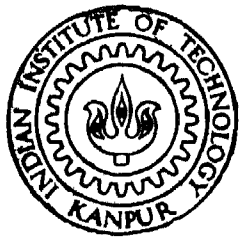
Implementation of GUI and Non-Blocking Queueing Network Analysis for QNAT

by

D M BHASKAR



Deptt of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JANUARY 1998

Implementation of GUI and Non-Blocking Queueing Network Analysis for QNAT

by
D M BHASKAR

Deptt of Electrical Engineering

Indian Institute of Technology, Kanpur

January, 1998

EE-1998-M-BHA-IMP

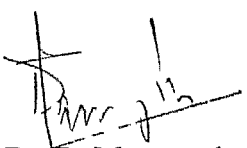
Entered in System

7/4-98

Certificate

This is to certify that this M Tech thesis work entitled *Implementation of GUI and Non Blocking Queueing Network Analysis for QNAT* has been carried out by D M Bhaskar under our supervision and has not been submitted elsewhere for a degree

Dr S K Bose
Professor
Department of Electrical Engineering
Indian Institute of Technology Kanpur



Dr D Manjunath
Assistant professor
Department of Electrical Engineering
Indian Institute of Technology Kanpur

Contents

	Page No
List of figures	(i)
List of tables	(ii)
Chapter 1 Introduction	01
Chapter 2 Queueing Network Models	05
2 1 Introduction	05
2 2 Queueing network with Multiple Classes of Customers	10
2 3 Compact Representation of Portions of a Queueing Network	13
2 4 Networks of GI/G/m Queues	18
2 5 Fork Join Queues	20
2 6 Conclusion	27
Chapter 3 Methods Used for Solving Networks with Infinite Capacity Queues	28
3 1 Queueing Networks with a Single Class of Jobs	28
3 1 1 Open Networks	28
3 1 2 Fork Join Analysis (Open Networks)	35
3 1 3 Closed Networks	38
3 1 4 Fork Join Analysis (Closed Networks)	42
3 2 Queueing Networks with Multiple Classes of Jobs	48
3 2 1 Open Networks	49
3 2 2 Closed Networks	49
3 2 3 Mixed Networks	54
Chapter 4 Summary and Future work	61
4 1 Summary	61
4 2 Future Work	62
Bibliography	63

Acknowledgements

I am grateful to my thesis supervisors Dr S K Bose and Dr D Manjunath who helped me to chose a problem which I felt would be challenging I sincerely thank them both for their excellent guidance and encouragement which helped me to get this thesis to the present state

I should be thankful to all the members in the Telematics and ERNET Labs with special mention to Mr Sanjeev Kumar for extending his co operation in solving my windows programming problem during my thesis work and Mr T Srinivas Rao senior research scholar for his valuable suggestions in memory handling in computer programs and UNIX environment Their casual interaction brings a lively environment in the lab

I should thank my senior M N Umesh for explaining his thesis and windows programming to me during his thesis work I also thank T Hema for her co operation and help extended to me during my thesis work

It is the extremely lively enthusiastic and encouraging environment provided by my classmates maheedhar murty hemachandra and sonia that made my work in the lab ever remembering

It is time for me now to mention about my friends sunil giri murali lakshman sateesh srinivas and others that made my academic stay ever remembering

Abstract

In this thesis we have implemented some well known techniques for solving queueing networks that have only non blocking nodes in the network in a user friendly software package called Queueing Network Analysis Tool (QNAT) We have also developed some techniques to reduce fork join queues that may or may not have synchronising queues and have made them available in QNAT QNAT can solve open and closed queueing networks with single class jobs and open closed and mixed networks with multiple class jobs QNAT does not allow jobs to change classes

For the analysis of open networks we use the technique developed for networks of GI/G/m queues by Whitt For closed networks we use the well known Mean Value Analysis (MVA) technique For mixed networks networks that have open and closed classes of customers we first find the utilisation of the servers for the customers of the open classes This utilisation is used to reduce the service rate (inflate the service demand) for the closed classes of customers We now consider only the closed classes of customers but with the reduced rate of service at the queues and then solve the network using the MVA technique This gives us the performance parameters for the closed classes of customers The performance of the open classes of customers is then obtained from those of the closed classes of customers

In QNAT fork join queues can also be specified in the network The fork join queue may or may not have synchronising queues after the service queue in the siblings The network may be open or closed In all these cases a flow equivalent service centre (FESC) for the fork join queue is obtained using suitable techniques The fork join queue is replaced by this FESC and the resultant network is solved using standard techniques Fork join queues with synchronising queues is not supported by QNAT for open networks QNAT can also handle queueing networks that have finite capacity queues and its implementation are detailed in a companion thesis

QNAT has a user friendly graphical user interface (GUI) based front end from which we can create a network save the network with suitable names open the networks that were previously saved print the solution results into a file etc QNAT also has an online help to guide the user at every stage The executable files have been developed in such a way that future modifications like providing a simulation option or adding any new capabilities can be easily incorporated

A beta version of QNAT is being distributed over the Internet It can also be obtained from the ERNet/Telematics Laboratory of IIT Kanpur

List of Figures

Figure2 1 An open queueing network 06

Figure2 2 A closed queueing network 07

Figure 2 3 Norton s reduction in electrical circuits 13

Figure 2 4 Norton s reduction in queueing networks 15

Figure 2 5 A fork join queue with *k* sibling queues 21

Figure 2 6 Fork Join without synchronizing queues 22

Figure 2 7 Fork Join with synchronozing queues 23

Figure 2 8 Fork join with synchronizing queues for closed network 25

Figure 2 9 State transition diagram of Fig 2 8c 26

Figure3 1 The open network for Example 3 1 34

Figure3 2 A fork join model without synchronizing queues in an open network for Example 3 37

Figure 3 3 Closed network with a single class of customers for Example 3 3 40

Figure3 4 Fork join queues without synchronizing queues in closed network for Example 3 4 43

Figure3 5 A fork join queue with synchronizing queues in closed network for Examples 3 5 and 3 6 47

Figure 3 6 Multiple class Closed Network(2 classes) for Example 3 7 53

Figure 3 7 Multiple classes in Mixed Network(2 classes) (one open class and one closed class) for Example 3 8 58

List of Tables

Table 3 1 Input parameters of Example 3 1 34

Table 3 2 Transition probability matrix of Example 3 1 34

Table 3 3 Output parameters of Example 3 1 35

Table 3 4 Input parameters of Example 3 2 38

Table 3 5 Output parameters of Example 3 2 38

Table 3 6 Input parameters of Example 3 3 41

Table 3 7 Routing matrix for Example 3 3 41

Table 3 8 Output parameters of Example 3 3 41

Table 3 9 Input parameters of Example 3 4 44

Table 3 10 Output parameters of Example 3 4 44

Table 3 11 Input parameters of Example 3 5 48

Table 3 12 Output parameters of Example 3 5 48

Table 3 13 Input parameters of Example 3 6 48

Table 3 14 Output parameters of Example 3 6 48

Table 3 15 Input parameters of Example 3 7 53

Table 3 16 Mean service time of Example 3 7 53

Table 3 17 Routing probabilities for class 1 customers of Example 3 7 54

Table 3 18 Routing probabilities for class 2 customers of Example 3 7 54

Table 3 19 Output parameters of Example 3 7 54

Table 3 20 Input parameters of open class in Example 3 8 59

Table 3 21 Input parameters of closed class in Example 3 8 59

Table 3 22 Routing probabilities for Example 3 8 59

Table 3 23 Output parameters of open class in Example 3 8 60

Table 3 24 Output parameters of closed class in Example 3 8 60

Chapter 1

Introduction

The design and development of many of the rapidly evolving complex systems requires that their throughput and delay performance be studied during the design and development process Examples of such systems include computer systems manufacturing systems networked communication systems and distributed database systems There are many ways to study the performance of such systems An obvious method would be to use intuition and perform trend extrapolation The use of such a method relies heavily on the experience of the performance analyst and the reliability associated with this is very low because of the absence of a scientific basis for the conclusions from the study Further there will always be very few people who can develop the ability and the intuition to accurately predict the performance of systems through trend extrapolation A second option available to performance analysts is that of experimental evaluation Here the actual system will be subjected to various load conditions and the relevant performance parameters measured This approach is obviously very expensive Further it might not be possible to obtain realistic conditions for the experimental setup A considerable amount of extrapolation may be necessary from the results of a few tests and the results obtained by this approach may not be easily generalised A third approach the one preferred by many performance analysts is that of analytical (or mathematical) modelling Among the many mathematical techniques that can be used to model systems let us consider queueing network based models

Many systems can be represented as a network of queues Jobs enter a queue wait for their turn to receive service at the queue and upon completion of service they may either enter another queue or leave the system The arrival process to a queue the service discipline at a queue the service process at a queue and the routing probabilities after

service at a queue are some of the parameters that specify the queueing network model of a system Under certain assumptions of the parameters specifying the queueing network an analytical solution may be obtained by evaluating a set of linear equations obtained from the queueing network specification If the analytical solution is not easily available a computer simulation model for the queueing network model of the system under study can be created The principal advantage of a simulation study is its flexibility However simulation models are generally expensive to obtain because of the need to write and debug complex computer programs

From the above discussion it is evident that analytical modelling is the preferred method in the performance analysis of systems In this thesis our interest is in queueing network based analytical modelling of systems

Queueing network models with simple assumptions achieve a favourable balance between accuracy and efficiency These models achieve relatively high accuracy at a relatively low cost queueing network modelling is a small subset of the techniques of queueing theory Much of queueing theory is oriented towards modelling a complex system using a single queue or a service centre with complex characteristics A detailed performance analysis of such systems can be obtained by using sophisticated analytical techniques Simulations can also be performed on such complex single queue representations of systems but the performance results obtained are based on complex assumptions and will generally be difficult to generalise In queueing network modelling a network of simple queues are considered and simple evaluation algorithms are used to obtain meaningful performance measures with an appropriate balance between accuracy and efficiency

Constructing a queueing network model involves the following steps definition of the system parameterization and evaluation Defining systems is relatively easy whenever these systems can be represented as a network of queues In most systems that interest us parameterization of the input variables that define the system and defining the output

variables of interest is relatively straightforward and system description is easily accomplished It is the solution of the model that requires extensive mathematical calculations Further if the system is to be evaluated for a number of different sets of input parameters the system evaluation becomes tedious and time consuming Thus there is a great need for a tool to solve queueing network and obtain their performance parameters

The steps in the specification and solution of the queueing network model for any system can be integrated into a single user friendly package where the simple terminology of the queueing network system rather than the language of queueing theory is used We are aware of the existence of a large number of software packages that have been developed to specify and solve queueing network systems Some of these are CADS (1978) BEST/1 (1980) CAPTURE/MVS (1982) RESQ (1982) MAP (1982) [1] PANACEA (1982) CMF/Model (1983) and QNA (1983) [ ] Most of these packages are not easily available Also in these packages the system specification is through a text file which in these days of GUI based user friendly computing would not be very useful Further all these packages have been designed to solve only a restricted class of queueing networks In 1996 an integrated user friendly package called NAST Network Analysis and Simulation Tool was developed in the ERNet/Telematics Laboratory of IIT Kanpur[3] This is similar in its capabilities to the packages mentioned above except that the input system and its parameters is specified through a GUI and the software is developed for Microsoft Windows™ on PCs and uses Mathematica™ as the compute engine In this thesis we have developed an updated version of NAST which can solve many new queueing models like BCMP queueing networks and queueing networks with fork join queues in addition to those supported by NAST This package is called QNAT (Queueing Network Analysis Tool) QNAT can also solve queueing networks that have finite capacity nodes

---

[1] Edward D Lazowska John Zahorjan G Scott Graham and Kenneth C Sevcik *Quantitative System Performance* Prentice Hall Inc 1984
Ward Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 198
[3] M N Umesh *A Window based Package for Queueing Network Analysis* M Tech Thesis IIT Kanpur Feb 1997

The following briefly summarises the contents of this thesis In Chapter 2 we give a brief overview of the queueing network models that we have incorporated into QNAT In Chapter 3 we provide the details of the algorithms used in this package We will not discuss the algorithms for solving queueing networks that have finite capacity nodes These will be discussed in a companion thesis[1] We also provide examples and verifications for the algorithms Chapter 4 contains an overview of the contents of the QNAT software package conclusion and outline for future work

[1] H Tahilramani *Solving Queueing Networks with Finite Capacity Nodes in QNAT* MTech Thesis In preparation

Chapter 2

Queueing Network Models

2 1 Introduction

Most complex systems that provide any kind of service at multiple nodes can be modeled as a network of queues by an appropriate definition for the arrival process the service process and the routing of the customers after service at a node Thus by evaluating the performance of the equivalent network of queues representation of the system we can study the performance of the original complex system Consider a machine shop example with a number of machines Jobs will enter the machine shop from outside They will receive service from the various machines in the shop in some possibly random sequence and finally leave the shop In the equivalent queueing network model the machines are represented by the individual queues providing service the jobs represented by the customers and the sequence of machines used by jobs corresponds to the routing process Under fairly general assumptions for the arrival service and routing processes the equivalent network of queues can be shown to be a multi dimensional birth death process

The first useful result available for queueing networks is Jackson s Theorem This theorem states that if the external arrivals into the queues form a Poisson process the service time at each of the queues is independent of the service at the other nodes and has an exponential distribution and if the routing is Markovian i e the probability of going from node i to node j after completion of service at node i is independent of the past then the joint distribution of the state vector of the system $\underline{n}=[n_1\ n_2\ n_3\ \ n_N]$ has a product form solution of the form

$$P(n_1\ n_2\ \ n_N) = \frac{P_1(n_1)\ P_2(n_2)\ \ P_N(n_N)}{G} \qquad \text{for } n=0\ 1\ 2 \qquad (2\ 1)$$

where $P(n_1\ n_2 \quad n_N)$ is the probability that the network of N queues is in state $\underline{n} = (n_1\ n_2 \quad n_N)$ n_i is the number of customers in queue i $P_i(n_i)$ is the probability of queue i having n_i customers and is calculated by treating node i independent of the other queues in the network and by obtaining the arrival rate into that queue from the arrival process into the network and the routing process G is the normalising constant to make the right hand side of the above equation a probability mass function

The following example illustrates Jackson s theorem Consider the simple network of two queues shown in figure 2 1 Jobs enter queue 1 with rate λ_1 After receiving service at queue 1 the customer comes back to node 1 for another service with probability p or goes to queue 2 with probability $(1\ p)$ Customers can also arrive to queue 2 with rate λ The service rates at nodes 1 and 2 are μ_1 and μ respectively

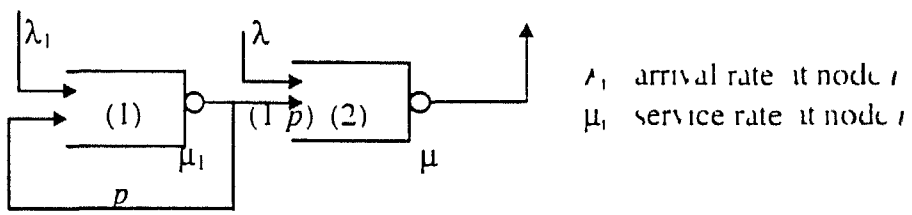


Figure2 1 An open queueing network

From Jackson s theorem

$$P(n_1\ n\ ) = P_1(n_1)\ P\ (n\ ) = \rho_1^{n1}\ \rho^{\ n}\ /\ G \quad \text{if } \rho_1 < 1 \text{ and } \rho\ < 1$$

where

$$\rho_1 = \frac{\lambda_1}{(1\ p)\ \mu_1} \qquad \rho_2 = \frac{\lambda_1 + \lambda}{\mu}$$

It can also be easily seen that the normalising constant G will be obtained from the normalising constants of the two M/M/1 queues $(1\ \rho_i)$ i e $G^{\ 1} = (1\ \ \rho_1)(1\ \ \rho\ )$

Statistics of the waiting times and the queue lengths at the queues can be easily obtained from the above equations

It has also been shown that if the network of queues contains a fixed number of customers at all times i e the network is a closed queueing network with no external arrivals into the network or departures from it the product form solution of Eqn 2 1 is retained In calculating $P(n_i)$ we have to consider a relative arrival rate rather than the absolute arrival rates into the node This is because the network is closed an absolute arrival rate cannot be defined Also the calculation of the normalisation constant G becomes a lot more difficult and was the main computational bottleneck in the useful application of this queueing model

Consider the example shown in figure 2 2 There are M customers in the network A customer after receiving service at queue 1 will either return to queue 1 for another service with probability p or go to queue 2 with probability 1 p After receiving service at queue 2 the customer goes to queue 1 with probability 1

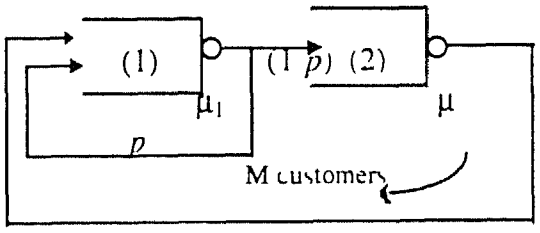


Figure2 2 A closed queueing network

The probability that the system is in state $\underline{n}= [n_1\ n_2]$ is given by

$$P(n_1\ n_2) = \frac{P_1(n_1)\,P_2(n_2)}{G} = \frac{\rho_1^{n_1}\,\rho_2^{n_2}}{G} \quad \text{for } n_1+n_2 = M \qquad (2\ 2)$$

$$= 0 \quad \text{for } n_1+n_2 \neq M$$

where $\rho_1 = \lambda_1 / \mu_1 = 1 / ((1\ p)\mu_1)$

$\rho_2 = \lambda / \mu_2 = 1 / \mu_2$

Here we have calculated the arrival rates relative to node 2 The normalising constant G is obtained by summing $P(\underline{n})$ over all possible values

$$G = \sum_{n1+n2\ M} P(\underline{n}) \qquad (2\ 3)$$

As can be seen from the above equation evaluating G requires the summation of the product terms over $^{M+N\ 1}C_N$ terms Further these terms have to be searched in a region of M^{N+1} terms This is clearly computationally inefficient and efficient algorithms are necessary for closed queueing network models to be usefully applied Also as we will now show queue length and waiting statistics of a network of N queues with M customers can be expressed as functions of the normalising constant of networks with V or fewer queues and M or fewer customers

Let λ_i be the relative arrival rate to node i μ_i the service rate at node i and the $X_i = \lambda_i/\mu$ λ_i is obtained by solving the equation $\lambda\underline{P}=\lambda$ where $\underline{P}$ the routing matrix containing the probabilities P_{ij} the probability of going to node j after service at node i $G(N\ M)$ is the normalising constant evaluated for a closed queueing network with N nodes and M customers The following can be easily shown

Utilisation at node $i = \rho_i = X_i\ G(N\ \ 1) / G(N)\ \ P(n_i > 1)$ (2 4)

Nett arrival rate to node $i = \lambda_i = \rho_i\ \mu_i$ (2 5)

Average number of customers at node $i = E(n_i) = \sum_{=1}^{N} (X_i)^n\ G(N\ 1) / G(N)$ (2 6)

Thus we see that the solution of a queueing network model boils down to the evaluation of a normalizing constant In the following we describe the convolution algorithm to evaluate $G(N\ M)$

$$P(n_1\ n_2 \qquad n_N) = (\ 1 / G(N)\ ) \prod_{=1}^{N} (X_i)^n \qquad (2\ 7)$$

The normalizing constant can be evaluated by following the recursive algorithm

```
Initialize  for m = 1 2      ,M G(0 m) =1 0
for n = 1 2      ,N
    for m = 1 2      ,M
        G(n m) = G(n m  1) + X_m G(n  1 m)
```

If the relative utilization s are larger than the actual utilization s the normalizing constant is greater than 1 and may be very large On the other hand if the actual utilization s are much larger than the relative utilization s the normalizing constant may become very small This can cause numerical overflow or underflow during the calculation of the normalizing constant To overcome this numerical difficulty an iterative technique known as Mean Value Analysis (MVA) has been developed This technique avoids the problem of numerical overflow or underflow by dealing with the ratios of the normalizing constant Also it has the advantage of directly computing the average performance measures The algorithm in this analysis is based on two fundamental theorems viz Little s Theorem and Mean Value Theorem From Little s theorem

$$N_i = T W_i \lambda_i$$

where N_i is the mean number of jobs at queue i T is the mean throughput of the network (on a reference queue) W_i is the mean delay at queue i and λ_i is the relative arrival rate at queue i The *Mean Value Theorem*[1] states that a customer arriving to a queue in a product form queueing network sees precisely the same distribution of customers as an outside observer would see if there were one less customer in the network From these two theorems a recursive algorithm is developed in which waiting time of a new customer is computed from the waiting times computed using one less customer in the network Thus starting with zero customers in the network we can iterate upto M The outline of the algorithm is given below and is discussed in detail in the next chapter In this algorithm τ_i is the mean service time at queue i

[1] M Reiser and S S Lavenberg *Mean Value Analysis of Closed Multichain Queueing Networks* J of ACM Vol 27 No 2 pp 313 322 April 1980

```
Initialize for i = 1 2      ,N  l_i(0) = 1 0
for m = 1 2      ,M
    for i= 1 2       ,N
        w_i = τ_i * l_i (m 1)
    T = m / Σ_{i=1}^{N} w_i * λ_i
    for i = 1 2      ,N
        l_i(m) = T * w_i
```

2 2 Queueing Networks with Multiple Classes of Customers

In the previous section we discussed networks with a single class of customers Further we limited ourselves to FCFS queues with exponential service time distributions It was first shown that a queueing network model for a central server system with a processor sharing CPU and exponentially distributed I/O service time had product form solution if the CPU service time distribution had a rational Laplace transform It was also shown that the steady state behaviour of such models depends only on the mean CPU service time and not on its higher moments These results were then extended to arbitrary networks by defining and using the concept of local balance for a queueing network to show that certain other queueing disciplines like processor sharing and last come first served with preemptive resume also satisfy local balance and hence have a product form solution Baskett Chandy Muntz and Palacios [1] extended these results to a large class of networks that serve multiple classes of customers They showed that the product form solution exists for all kinds of networks open closed or mixed if they satisfy the following conditions

- All the service centres have any one of the following four service discipline first come first served (FCFS) last come first served with preemptive resume (LCFS PR) processor sharing (PS) and an infinite server queue (IS)
- While waiting or receiving service at a queue the customers are not allowed to change classes After service completion they may change classes and

[1] F Baskett K M Chandy R R Muntz and F G Palacios *Open Closed and Mixed networks of queues with different class of customers* J of ACM Vol 22 No 2 pp 248 260 April 1975

should be routed according to fixed probabilities Routing is Markovian and does not depend on the previous visits of the customer

- At a FCFS queue the service time distribution must be identical and exponential for all classes of jobs At other types of queues the service times should have probability distributions with rational Laplace transforms and different classes of jobs may have different distributions
- At a FCFS queue the service time can depend only on the total queue length at that queue At LCFS PR PS and IS queues the service time for a given class of customers can depend only on the total queue length at that queue of that class but not on the queue length of other classes
- In an open network the interarrival time should be exponentially distributed Bulk arrivals are not permitted The arrival rate can be state dependent
- A network can be open with respect to some classes and closed with respect to other classes of customers Such networks are called mixed networks

Queueing networks satisfying the above criteria are called as BCMP networks It has also been shown that a queuing network that satisfies the $M \Rightarrow M$ property has a product form solution A queue is said to have $M \Rightarrow M$ property if a Poisson input process results in a Poisson output process

We now explain the allowable service disciplines in detail

- First Come First Served (FCFS) Under FCFS scheduling customers are served in the order in which they arrive The allowable service time distribution is negative exponential The service rate can be state dependent in which case $\mu_i(j)$ will be the service rate at queue i when there are j customers in that queue In modelling a computer system an example use of the FCFS queue would be to represent a disk containing user files for a number of classes Since the basic operations performed at the device by the various classes are the same it is reasonable to assume that the average service times across classes are nearly equal The actual number of file accesses for

customers of each class can be represented in the model by appropriate values of the visit ratios of each class at each queue

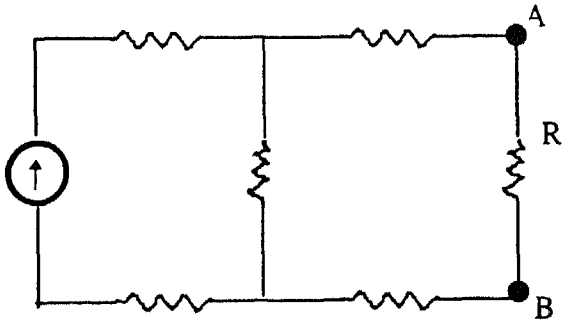
- Last Come First Served with Preemptive Resume (LCFS PR) Under this discipline an arriving customer preempts the customer in service (if any) and its service is begun immediately When a customer completes service the most recently preempted customer resumes service at the point at which it was interrupted The service time distributions should have a rational Laplace transform This kind of service discipline can be used to model a CPU in a system where high priority system tasks are dispatched more frequently
- Processor Sharing (PS) This is the idealization of round robin (RR) scheduling Under this discipline the control of the processor circulates among all jobs in the queue Each customer receives a quantum of service before it must relinquish control to the next customer in the queue and rejoin the queue at its tail The service time distributions should have a rational Laplace transform This kind of service discipline can be used to model a CPU scheduling in multiprogrammed computer systems where some form of round robin scheduling is employed
- Infinite Server (IS) This is also called delay service discipline There will be simultaneous service for all the customers in the queue by an individual server and hence the total delay in such a queue is equal to the service time at the queue Again the service time distributions should have a rational Laplace transform

It has been explained earlier that FCFS queues are allowed to have a state dependent service rate An IS queue can also be thought of as a queue with state dependent service For example in a single class queueing network with a service rate of μ for each customer the IS queue will have a service rate of $n\mu$ when there are n customers in it The most important use of a queue with state dependent service is in the representation of Flow Equivalent Service Centres (FESCs) that can be used to reduce certain parts of the network and represent them as a single queue in a smaller possibly simpler queueing

network representation of the system The FESCs can also be studied in isolation In the next section we will discuss the compact representation of portions of a queueing network using FESCs

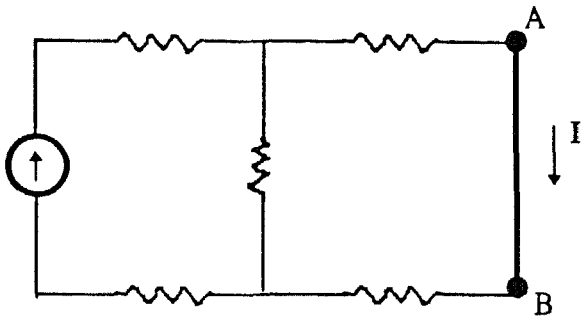
2 3 Compact Representation of Portions of a Queueing Network

Divide and Conquer is a well known technique to solve complex problems In electrical circuits by applying Norton s theorem any passive electrical circuit between two terminals can be replaced by an equivalent current source and a resistance in parallel to the current source The value of the current source is equal to the current that would flow when the two terminals are shorted in the original network The parallel resistance is determined by determining the resistance of the network when looking into it from the terminals This is illustrated in the example in Figure 2 3 We can study the electrical network shown in Figure 2 3a for various values of R by replacing the remaining components of the circuit by a single current source whose current capacity I is determined by shorting the points A and B (desired resistor R) and finding the current through this short circuit This is shown in Figure 2 3b The parallel resistance R_p is determined from Figure 2 3c Thus the electrical network of Figure 2 3a can be replaced by the equivalent shown in Figure 2 3d

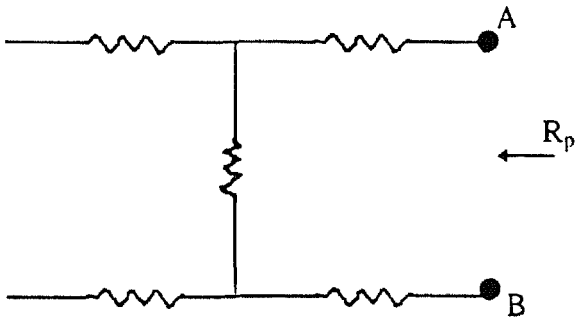


a Original Network

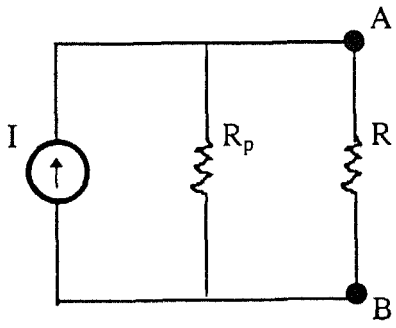
Figure 2 3 Norton s reduction in electrical circuits



b Network to evaluate the current source

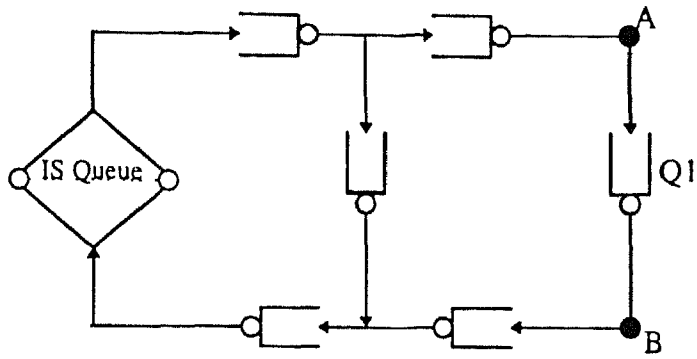


c Network to evaluate the parallel resistance

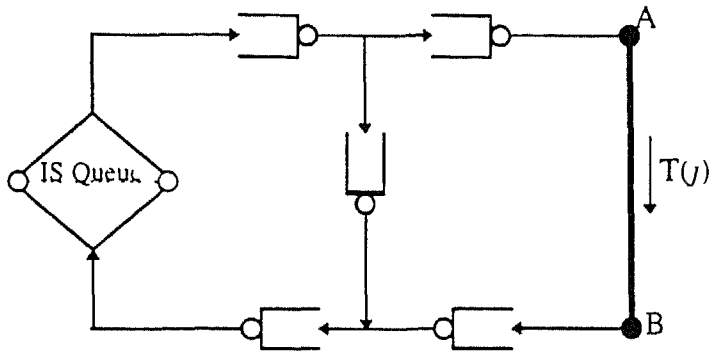


d Equivalent Network

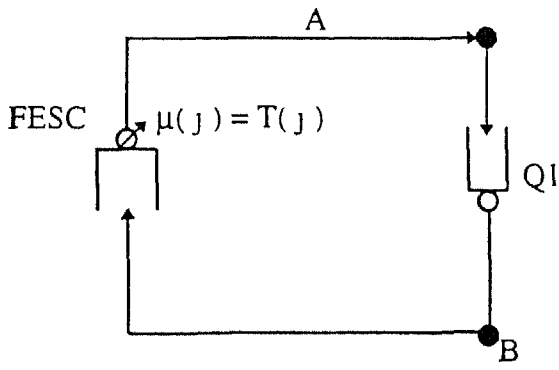
Figure 2 3 Norton s reduction in electrical circuits (contd)



a Original network



b Network to obtain the FESC J is population of the network



c Equivalent network

Figure 2 4 Norton s reduction in queueing networks

IChandy Herzog and Woo [1 2] described a method similar to Norton s theorem for reducing queueing networks They describe a technique to reduce a queueing network with N FCFS exponential service queues and M customers in the network In this technique a smaller network in which all queues except those in the subnetwork are replaced by a single FES They show that for certain classes of system parameters the behaviour of the equivalent network is the same as that of the original network This result is called the Chandy Herzog Woo Theorem The technique is illustrated below

Consider the queue Q1 between points A and B of Figure 2 4a Obtaining the queueing statistics at this queue for different characterisations of it will be simplified if the rest of the network can have a more compact representation preferably as a single queue To achieve this we can construct an equivalent queue representation of the rest of the network The method is known as Norton reduction and is described below The subnetwork is replaced by a state dependent queue with service rate $\mu(j)$ which is a function of the number of jobs j in the queue This service rate $\mu(j)$ is the same as the throughput $T(j)$ between the points A and B of Figure 2 4b when there are j customers in the network Note that the network of Figure 2 4b is the same as that in Figure 2 4a with a short across the queue Q1 The equivalent network representation is shown in Figure 2 4c In general the portion of the network that can be isolated is not limited to a single queue and can be a network of queues Such a network of queues is called the designated netwrok and the remaining network is called the aggregate The aggregate is reduced to a FESC and the performance of the designated network is studied for the variation of the parameters of the designated network

The hierarchical decomposition described above produces exact results for the large class of BCMP networks The decomposition method based on the Chandy Herzog Woo theorem is summarized below

---

[1] K M Chandy U Herzog and L Woo *Parametric Analysis of Queueing Networks* IBM Jour of Res and Develop 19 36 Jan 1975

K M Chandy U Herzog and L Woo *Approximate Analysis of General Queueing Networks* IBM Jour of Res and Develop 19 36 Jan 1975

- Select the designated subnetwork of queues from the original network that needs to be studied in detail The remaining portion of the network is the aggregate and will be reduced to a single FESC
- Generate a queueing network in which the service times at all the queues in the designated network is set to zero i e they are shorted Note that the designated network should be selected such that the throughput through all the shorts in the new network obtained above should be identical
- Solve the above queueing network using known techniques Solve this for all possible values of the network population in the original network The throughput through the short for different populations corresponds to the service rate of the FESC with that many in its queue
- The FESC for the aggregate is now available
- Obtain the equivalent network in which the designated network is in place and the aggregate is replaced by its FESC The queueing statistics of the designated network in the equivalent network will be identical to those in the original network

Aggregation produces exact results for queueing networks with a product form solution However aggregation of product form networks may turn out to be computationally more expensive than simple MVA technique if the purpose is to solve for only one set of parameters of the designated network The real advantage of aggregation is in solving non product form networks The components that do not have product form are put in the designated network and a FESC is obtained for the aggregate which contains only component queues that yield a product form solution The equivalent can be solved by using approximate non product form solution techniques or by simulation Since solving an actual non product form network is computationally very expensive the reduced number of queues in the equivalent model would reduce the computing time However in this case aggegation is only an approximation This is because the behaviour of the aggregate cannot be exactly replaced by the FESC since the information regarding the location of the customers in the aggregate is discarded However in many cases of

practical interest it turns out that the results of the actual network and the equivalent network are close enough and the aggregation is a useful approximation

2 4 Networks of GI/G/m Queues

So far we have been dealing with queues with Poisson external arrivals and service time distributions that were either exponential or had a rational Laplace transform Further it was only the mean service time that was used in the analysis We now consider queues where the external arrivals do not necessarily form a Poisson process and can have a general distribution for the interarrival times We also allow the service times to be characterised by a general distribution An example of a network of such queues would be a packet switched communication network where packet arrivals form a bursty process not necessarily Poisson and the service times for each packet are fixed Such networks can be modeled as a networks of GI/G/m queues The theory and algorithms for open networks of GI/G/m queues has been developed by Whitt[1] This algorithm requires that the first two moments of the interarrival time and service time distributions be known This technique can be used to solve an open network of queues having FCFS service discipline and multiple servers for single and multiple classes of customers The fundamental assumption made in this algorithm is that queues in the network are stochastically independent and a product form solution can be given The routing matrix is used to obtain the two moments of the service time and interarrival time distributions into the individual queues Each of the queues is then treated as an independent GI/G/m queue and a product form solution for the network is obtained

A customer that is in the network can experience any of the following three basic operations merging or superposition thinning or splitting and the departure of the customer from the network A multiplication factor associated with a queue γ determines

[1] W Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983
W Whitt *Performance of the Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983

if the customers are to be split ($\gamma > 1.0$) merged ($0.0 < \gamma < 1.0$) or not modified ($\gamma = 1.0$) at that queue. The routing matrix determines the next destination of a packet after its service at a queue.

In the following we briefly summarize the theory of the technique used in the Whitt algorithm used to analyse a network of GI/G/m queues. Variability of a distribution is defined as the ratio of the variance to the square of the mean for that distribution. The arrival process of customers to any queue is a point process. In this technique we will approximate the point process corresponding to the arrivals by an equivalent renewal process using two parameters rate (mean number of arrivals per unit time) and the variability (defined above). The approximation for a stochastic point process by a renewal process is described by *Whitt* [1]. This approximation involves the decomposition of the model by replacing the component flows (point process) by an independent renewal process. The distribution of the renewal process is obtained by fitting a distribution to a few moments of the interval between two point arrivals. Thus any point process is approximated by a renewal process characterized by the mean and squared coefficient of variance of the distribution that is fit to the moments evaluated from the properties of the point process. (Squared coefficient of variance is the variability parameter.)

The Whitt algorithm converts the resulting process due to merging, splitting and departure of traffic at the queues to a suitable renewal process (nett arrival process) characterized by its mean and squared coefficient of variance. This conversion is done by solving a set of linear equations for each parameter. The queueing network is then decomposed into individual queues which are analysed separately. Each queue is characterized by a standard GI/G/m queue with four parameters the mean and the squared coefficient of variance of the arrival process and the service time distributions. Approximate two moment formulae are available to compute the performance of these queues in literature. The decomposition into independent queues can be interpreted as a

[1] W. Whitt, *Approximating a Point Process by a Renewal Process: I. Two Basic Methods*, Operations Research, Vol. 30, No. 1, pp. 125-147, Jan.-Feb. 1982.

generalization of the product form solution for the *Markovian* networks In *Markovian* models the population vector at each node are stochastically independent so that the probability mass function of the vector is a product of the mass functions of the individual components of the vector The Whitt algorithm is described in more detail in the next chapter

2 5 Fork-Join Queues

There are situations when a given job is decomposed and distributed to be serviced in parallel and are aggregated after the service completion of the decomposed jobs Such queues are called fork join queues where the fork process corresponds to the decomposition and service in parallel and the join process corresponds to the aggregation after completion of all the sub jobs A fork join queueing system contains multiple parallel queues and can operate with or without synchronizing queues A synchronising queue is a waiting place for each of the parallel queues (sibling queues) where the customers wait for aggregation after receiving service at that queue If there is no synchronising queue the customers will wait at the server and block the server for other customers in that sibling queue In general the sibling and synchronising queues in a fork join node will have infinite queueing capacities use FCFS service discipline and operate independently A fork join queue is illustrated in Figure 2 5 An incoming job is split at forking node A and aggregated at the joining node B Every job arriving at the forking node consists of the same number of subtasks as the number of sibling queues and upon arrival at the forking node is immediately divided into a number of subtasks and each is simultaneously submitted to the sibling queues

An exact analysis of a fork join queue is difficult However many approximate models have been described in literature and we have used the techniques described in [1 3]

---

[1] C Kim and A K Agrawala *Analysis of the Fork Join Queue* IEEE Trans on Computers Vol 38 No 2 pp 250 255 Feb 1989
Y C Liu and H G Perros *A Decomposition for the Analysis of a Closed Fork/Join Queueing System* IEEE Trans on Computers Vol 40 No 3 pp 365 370 Mar 1991

[3] Y C Liu and H G Perros *Approximate Analysis of a Closed Fork/Join model* European Journal of Operational Research 53 pp 382 392 1991

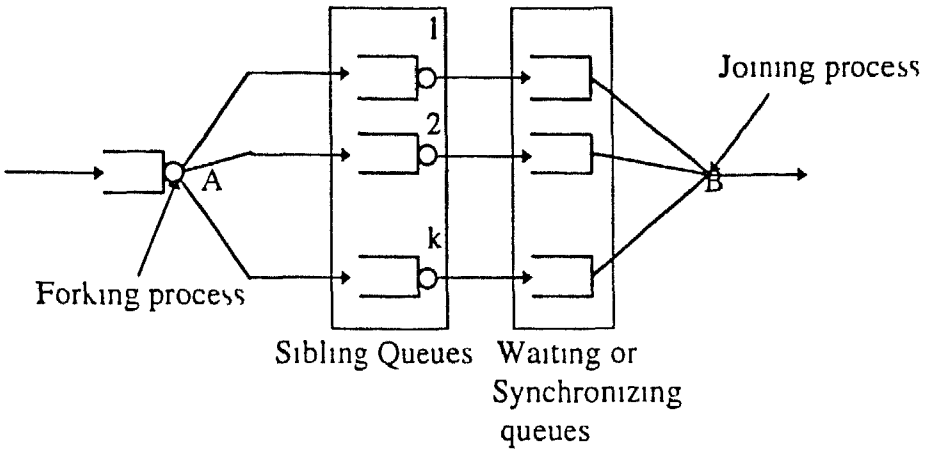
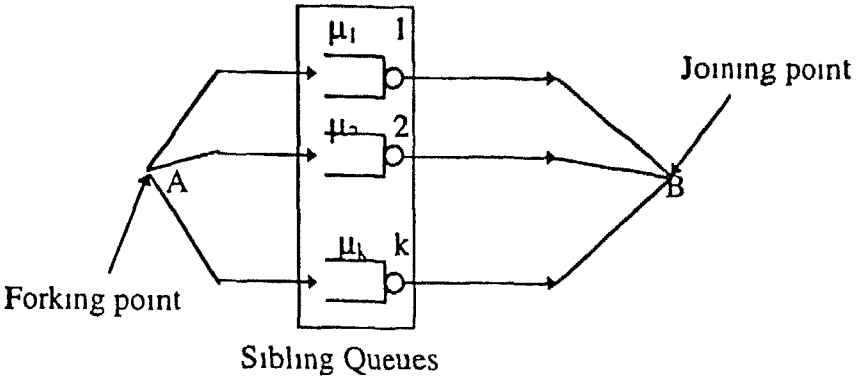


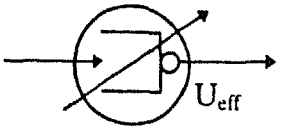
Figure 2 5 A fork join queue with k sibling queues

A fork join queue may or may not have a synchronising queue If there is no synchronising queue the subtasks wait at the server and they are rejoined after the last task has finished its service from the sibling queues Thus if there are k sibling queues they can be replaced by an equivalent queue whose service time distribution is equal to the distribution of the maximum of the k service time distributions of the sibling queues For QNAT we have assumed exponential service time distribution at the siblings not necessarily identical If the network is open we replace the fork join queue by an equivalent queue whose service time distribution is the distribution maximum of the k exponentials If the network is closed we first reduce the fork join queue to a single equivalent FCFS queue with the service time distribution given by the distribution of the maximum of k sibling service times We then fit an exponential distribution to this distribution that has the minimum mean square error Fork join queues without synchronising queues are illustrated in Figure 2 6

A fork join queue usually has a synchronising queue so that the subtasks can leave the sibling servers asynchronously The subtasks after completion of service at the sibling queues wait in the synchronizing queues When all the subtasks finish service they recombine and leave the queue In this kind of a queue the time spent in the



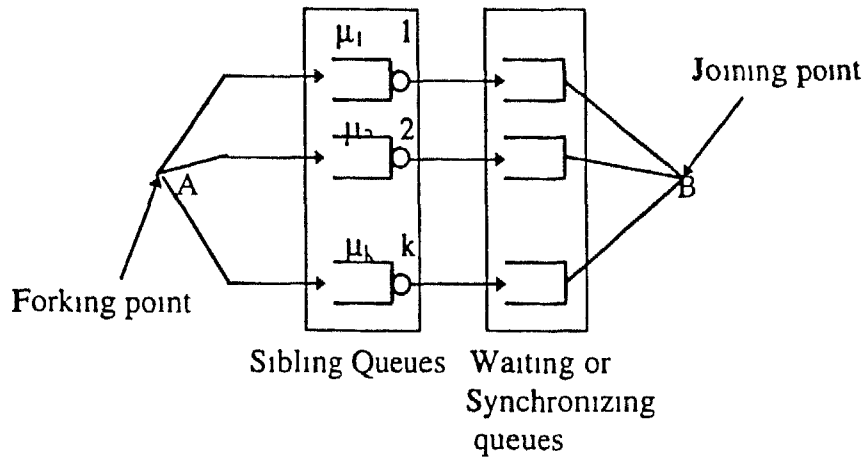
a A fork join queue without synchronizing queues



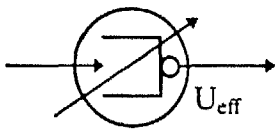
b Equivalent reduced queue

Figure 2 6 Fork Join without synchronizing queues

synchronising queue waiting for completion of the subtasks can be a significant portion of the response time (The response time for a job is the time between the forking and the joining instants of the job) Liu and Perros have developed an algorithm for the analysis of a closed queueing network with synchronizing queues The algorithm is an iterative approximation and for a k sibling fork join queue with M customers it works as follows Obtain a two sibling closed fork join queue by choosing any two of the siblings from the original k This fork join queue with j customers can be solved to obtain the throughput for $j=1$ M This is used to obtain the Norton equivalent for the two sibling fork join queue Another two sibling queue is now obtained in which one of the siblings is the Norton equivalent from above and the other is one of the remaining siblings in the original k sibling queue This can be solved as before and the process repeated till all the k siblings have been used to obtain the Norton equivalent of the original k sibling fork join queue A more detailed description of the algorithm follows and the implementation details are given in the next chapter



a The fork join queue



b Equivalent reduced queue

Figure 2 7 Fork Join with synchronozing queues

Consider the closed queueing network with M customer and a k sibling fork join queue shown in Figure 2 8a To apply Norton s theorem between points A and B of Figure 2 8a the network shown in Figure 2 8b is used To apply Norton s theorem consider sibling queues 1 and 2 as shown in Figure 2 8c This system can be solved exactly as follows Let P_{ij} be the probability that there are i jobs in sibling queue 1 and j jobs in sibling queue 2 This is a closed queueing network with state dependent service rates at the queues The state transition diagram for M=3 is shown in Fig 2 9 In this figure (i,j) represents the state of the two queue system and corresponds to having i customers in sibling queue 1 and j customers in sibling queue 2 It is easy to see that states (0 2) (1 2) (2 2) (0 1) (1 1) (2 1) (0 0) (1 0) and (2 0) are not possible Thus the total number of feasible states are $(M+1)^2 - M^2$

From the state transition diagram of Figure 2 9 for $M = 3$ we have

$$P_{M,M}(\mu_1(M) + \mu_2(M)) = P_{M,M-1}\mu_1(M) + P_{M-1,M}\mu_2(M)$$

$$P_{M,n}\,(\mu_1(M)+\mu_2(n)) = P_{M,n-1}\,\mu_1(M) + P_{M,n+1}\,\mu_2(n+1) \qquad 1 \le n \le M-1$$

$$P_{M,0}\,\mu_1(M) = P_{M-1}\,\mu_2(1) \qquad (2.8)$$

$$P_{n,M}\,(\mu_1(n)+\mu_2(M)) = P_{n+1,M}\,\mu_1(n+1) + P_{n-1,M}\,\mu_2(M) \qquad 1 \le n \le M$$

$$P_{0,M}\,\mu_2(M) = P_{1,M}\,\mu_1(1)$$

$$P_{M,M} + \sum_{n=0}^{M-1} P_{M,n} + \sum_{j=0}^{M-1} P_{j,M} = 1.0 \qquad (2.9)$$

Solving this set of equations we get

$$P_{n,M} = P_{M,M} \prod_{j=n+1}^{M} \rho_2(j) \qquad 0 \le n \le M-1$$

$$P_{M,n} = P_{M,M} \prod_{j=n+1}^{M} \rho_1(j) \qquad 0 \le n \le M-1 \qquad (2.10)$$

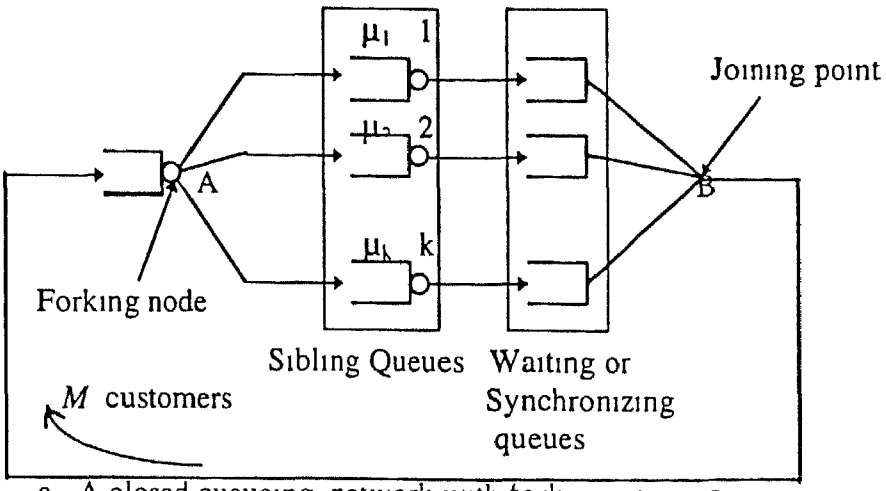
$$P_{M,M} = \frac{1.0}{1 + \sum_{n=0}^{M-1} \prod_{j=n+1}^{M} \rho_2(j) + \sum_{k=0}^{M-1} \prod_{l=k+1}^{M} \rho_1(l)} \qquad (2.11)$$

where

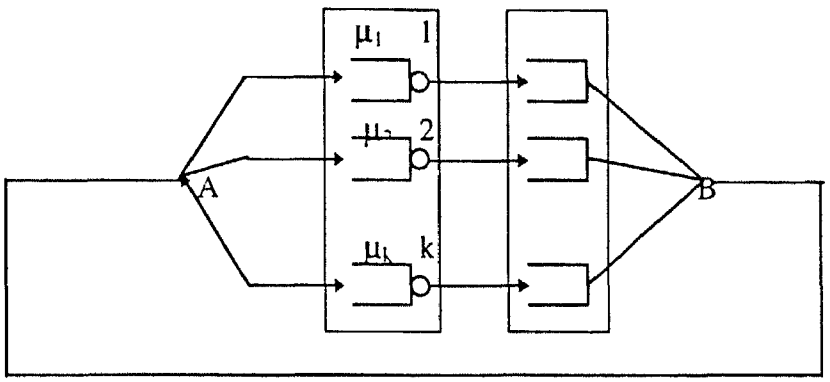
$$\rho_1(j) = \frac{\mu(j)}{\mu_1(M)} \qquad \rho(j) = \frac{\mu_1(j)}{\mu(M)}$$

The system throughput of Figure 2.8c is given by

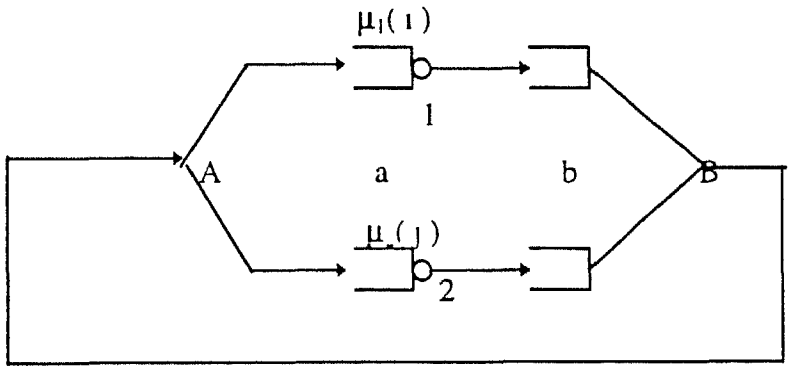
$$U_{eff}(M) = \mu_2(M) \sum_{n=0}^{M-1} P_{1,M} + \mu_1(M) \sum_{j=0}^{M-1} P_{M,j} \qquad (2.12)$$



a A closed queueing network with fork join queues



b The k sibling fork join queue as a closed queueing network



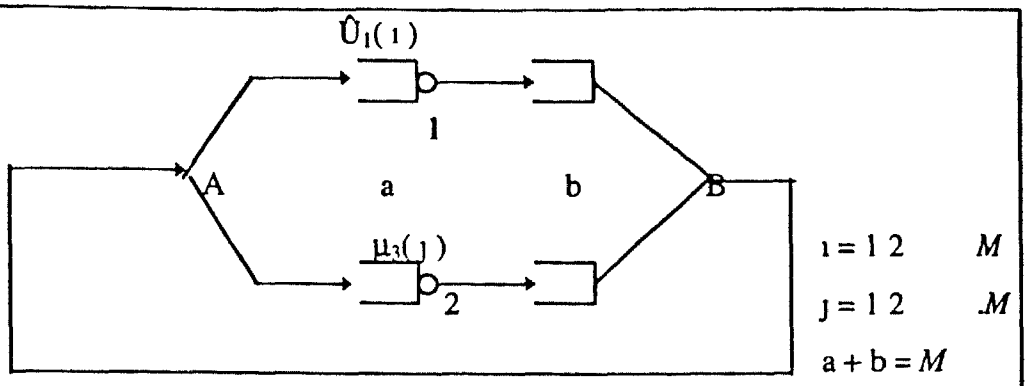
$i = 1\ 2 \quad M$

$j = 1\ 2 \quad M$

$a + b = M$

c Siblings queues 1 and 2 are reduced to an FESC with rate $_1(j)$

Figure 2 8 Fork join with synchronizing queues for closed network



d Sibling queue 3 and FESC from the reduction of siblings 1 and 2 are used to obtain the FESC for a three sibling fork join queue
The service rate of this FESC is ,

Figure 2 8 Fork join with synchronizing queues for closed network (contd)

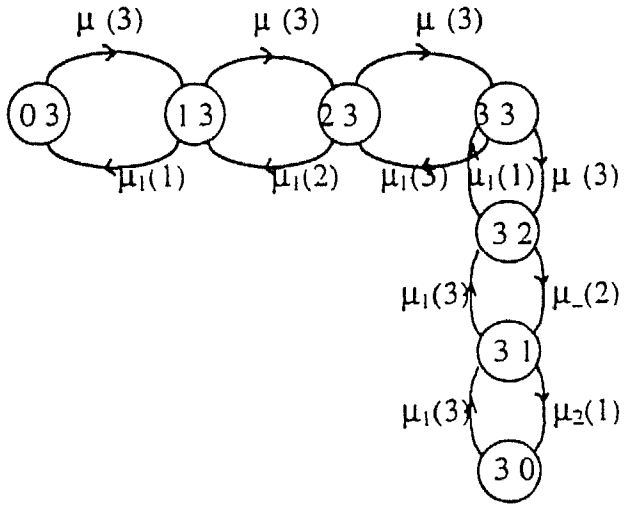


Figure 2 9 State transition diagram of Fig 2 8c

This Norton equivalent can be used with a third sibling queue of the k sibling queue to obtain another equivalent queue and this is shown in Figure 2 8d Thus by applying this technique iteratively k 1 times the k sibling queue finally reduced to a single queue with

state dependent service rate The implementation details of this algorithms is given in the next chapter

2 6 Conclusion

In this chapter we have explained the theory and the outline of the various algorithms that have been used in QNAT We started with a brief introduction to product form solutions to open and closed queueing networks Calculation of the normalising constant is the first bottleneck in the use of the closed queueing networks in modeling practical systems A convolution algorithm to calculate the normalising constant was given Mean value analysis a technique developed to overcome the numerical difficulties in the calculation of the normalising constant was then described Multiple classes of customers can be handled through BCMP networks We also described techniques to reduce parts of a large complex queueing and represent it by a flow equivalent server Fork join queues and approximate algorithms to analyse them were also described We also discussed networks of non Markovian queues

In the next chapter we will outline the algorithms that we have used in QNAT to solve BCMP queueing networks networks with fork join nodes and open networks of non Markovian queues We also give examples to use QNAT and explain the input mechanism to QNAT

Chapter 3

Methods Used for Solving Networks with Infinite Capacity Queues

In this chapter we describe the algorithms used in the QNAT package and provide examples to illustrate the use of the package and also to verify the working of the algorithm implementation For each type of analysis technique available through QNAT we will give a brief introduction to the technique describe the input and output parameters present the algorithm and then show some examples to explain the use of the method and also to verify the correctness of the algorithm implementation

3 1 Queueing Networks with a Single Class of Jobs

In these cases there will only be one class of customers (jobs) in the network i e all the customers in the network are treated to be of the same class The network can be either open or closed for this class of customers

3 1 1 Open Networks

In open networks the jobs come from the external world (i e outside the network) into the network and after getting service at different queues i e nodes in the network they depart to the external world In an open network there will be at least one node where jobs from the external world will enter the network The algorithm used for the analysis of an open queueing network with single class input is the GI/G/m method [1] This method requires the first and second moments of the interarrival time and service time distributions It computes the first and second moments of the net arrival process at each

[1] W Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983
W Whitt *Performance of the Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983

node in the network by solving a set of linear equations The performance of the network is then obtained by analyzing the individual queues in isolation Each queue is a standard GI/G/m queue characterised by the first two moments of the arrival process and the service times Details on this method may be obtained from [1]

Assumptions

- All the queues in the network have infinite buffer capacity
- The service discipline at all the queue in the network is FCFS
- There can be creation or combination of jobs at the queues in the network
- The routing matrix for the jobs in the network is static

Input Parameters

N number of nodes in the network

m_i number of servers at node i for $i = 1\ 2 \quad N$

λ_{0i} mean of the external arrival rate distribution at node i for $i = 1\ 2 \quad N$

C_{0i}^2 squared coefficient of variance of the interarrival time distribution at node i for $i = 1\ 2 \quad N$

τ_i mean of the service time distribution at node i for $i = 1\ 2 \quad N$

C_{si}^2 squared coefficient of variance of service time distribution at node i for $i = 1\ 2 \quad N$

γ_i multiplication factor at node i for $i = 1\ 2 \quad N$

[P] p_{ij} routing matrix for $i\ j = 1\ 2 \quad N$

C Squared coefficient of variance (SqCVar) $= \dfrac{\text{variance}}{\text{mean}^2}$

The multiplication factor γ_i at node i specifies the creation or combination of jobs at that node If $\gamma_i > 1$ then jobs are created if $\gamma_i < 1$ then jobs are combined and if $\gamma_i = 1$ then jobs are neither created nor combined

[1] M N Umesh *A Window based Package for Queueing Network Analysis* M Tech thesis IIT Kanpur Feb 1997

If λ_j is the nett arrival rate at node j then departure out of node j will be $\lambda_j \gamma_j$ For example consider a switch in a packet communication network After processing a packet at the switch the packet can be split into a number of smaller packets and then transmitted to other nodes in the network At the destination these packets can be recombined to get the original packet

Output Parameters

λ_i mean nett arrival rate at node i for $i = 1\ 2 \quad N$

C_{ai}^2 squared coefficient of variance of nett arrival rate distribution at node i for $i = 1\ 2 \quad N$

EN_i mean number at node i for $i = 1\ 2 \quad N$

C_{mi}^2 squared coefficient of variance of number at node i for $i = 1\ 2 \quad N$

EW_i mean waiting time at node i for $i = 1\ 2 \quad N$

C_{wi}^2 squared coefficient of variance of waiting time at node i for $i = 1\ 2 \quad N$

dep_i departure rate at node i for $i = 1\ 2 \quad N$

V_i visit counts at node i for $i = 1\ 2 \quad N$

U_i utilisation at node i for $i = 1\ 2 \quad N$

Algorithm 3 1 GI/G/m solution technique for open networks [1]

$$\tau_i = \frac{\tau_i}{1 \quad p_{ii}} \qquad \text{for } i = 1\ 2 \quad N$$

$$C_{si}^2 = C_{si}^2 + p_{ii}\,(1 \quad C_{si}^2) \qquad \text{for } i = 1\ 2 \quad N$$

$$\lambda_i = \lambda_{0i} + \sum_{j=1}^{N} \lambda_j p_{ji} \qquad \text{for } i = 1\ 2 \quad N$$

$q_{ij} = \gamma_i \lambda_i p_{ij}$ = arrival rate from node i to node j for $i\ j = 1\ 2 \quad N$

[1] W Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983
M N Umesh *A Window based Package for Queueing Network Analysis* M Tech thesis IIT Kanpur Feb 1997

(Algorithm 3 1 continued)

$$U_i = \frac{\lambda_i \tau_i}{m_i} \qquad \text{for } i = 1\ 2 \quad N$$

$$C_{ai}^{\,2} = \mu_i + \sum_{j\ 1}^{N} C_{aj}^{\,2}\, \nu_{ji} \qquad \text{for } i = 1\ 2 \quad N$$

where

$$\mu_i = 1 + \omega_i \left[(q_{0i}\, C_{0i}^{\,2} \quad 1) + \sum_{j\ 1}^{N} q_{ji} \{ (1 \quad p_{ji}) + \upsilon_j\, p_{ji}\, U_j^{\,2}\, x_j \} \right] \text{ for } i = 1\ 2 \quad N$$

$$\nu_{ji} = \omega_i\, p_{ji}\, q_{ji}\, \upsilon_j\, (1 \quad U_j) \qquad \text{for } i\ j = 1\ 2 \quad N$$

$$x_j = 1 + m_j^{\ 0\ 5} [\max (C_{sj} \quad 0\ 2) \quad 1] \qquad \text{for } j = 1\ 2 \quad N$$

$$\omega_i = \frac{1\ 0}{(1 + 4\,(1 \quad U_i^{\,2})\,(\upsilon_i \quad 1))} \qquad \text{for } i = 1\ 2 \quad N$$

$$\upsilon_j = \frac{1\ 0}{\sum_{i=1}^{N} q_{ij}} \qquad \text{for } j = 1\ 2 \quad N$$

$$q_{ji} = \frac{\lambda_{ji}}{\sum_{k=1}^{N} \lambda_{jk}} = \frac{q_{ji}}{\lambda_j} \qquad \text{for } i\ j = 1\ 2 \quad N$$

$$EN_i = U_i\, m_i + \lambda_i\, EW_i \qquad \text{for } i = 1\ 2 \quad N$$

$$C_{mi}^{\,2} = m_i\, U_i + U_i\, Pq_i \left(\frac{1 + U_i \quad U\ Pq_i}{(1 \quad U_i)^{\,2}} + m_i \right) \qquad \text{for } i = 1\ 2 \quad N$$

(Algorithm 3 1 continued)

where

$$Pq_i = Pz_i \left(\frac{(m_i U_i)^{?}}{m_i ! (1 \quad U_i)} \right) \qquad \text{for } i = 1\ 2 \qquad N$$

$$Pz_i = \frac{1\ 0}{\left(\sum_{k=0}^{-1} \frac{(m_i U_i)^{k}}{k !} \right) + \frac{(m_i U_i)^{m_i}}{m_i ! (1 \quad U_i)}} \qquad \text{for } i = 1\ 2 \qquad N$$

$$EW_i = \begin{cases} \left(\frac{C_{ai} + C_{si}^{?}}{2} \right) W_i^{M/M/m} & \text{if } m_i > 1 \\ W_i^{GI/G/1} & \text{if } m = 1 \end{cases} \qquad \text{for } i = 1\ 2 \qquad N$$

$$W_i^{M/M/m} = \frac{Csa_i \tau_i}{(1 \quad U_i) m_i} \qquad \text{for } i = 1\ 2 \qquad N$$

$$Csa_i = Pzero_i \frac{(\lambda_i \tau_i)^{n_i}}{m_i ! \left(1 \quad \frac{\lambda_i \tau_i}{m_i}\right)} \qquad \text{for } i = 1\ 2 \qquad N$$

$$Pzero_i = \frac{1\ 0}{\left(\sum_{k\ 0}^{1} \frac{(\lambda_i \tau_i)^{k}}{k !} \right) + \frac{(\lambda_i \tau_i)^{m_i}}{m_i ! \left(1 \quad \frac{\lambda_i \tau_i}{m_i}\right)}} \qquad \text{for } i = 1\ 2 \qquad N$$

(Algorithm 3 1 continued)

$$W_i^{GI/G/1} = \frac{\tau_i U_i (C_{ai}^2 + C_{si}^2) \beta_i}{2(1 - U_i)} \qquad \text{for } i = 1\ 2 \quad N$$

$$\beta_i = \begin{cases} e^{\left(\frac{2(1 - U_i)(1 - X_{ai}^2)^2}{3 U_i (C_{ai}^2 + C_{si}^2)}\right)} & \text{if } X_{ai}^2 < 1 \\ 1\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1\ 2 \quad N$$

$$C_{ii} = \frac{Pq_i (2 - Pq_i) \tau_i^2}{[m_i (1 - U_i)]^2 EW_i} \qquad \text{for } i = 1\ 2 \quad N$$

$$dep_i = \lambda_i V_i \left(1 - \sum_{j=1}^{N} p_{ij}\right) \qquad \text{for } i = 1\ 2 \quad N$$

$$V_i = \frac{\lambda_i}{\sum_{j=1}^{N} \lambda_{0j}} \qquad \text{for } i = 1\ 2 \quad N$$

Network Performance

$$\text{Throughput} = \sum_{j=1}^{N} \lambda_{0j}$$

$$\text{Total departure rate} = \sum_{i=1}^{N} dep_i$$

$$\text{Mean number} = \sum_{i=1}^{N} EN_i$$

$$\text{Squared coefficient of variance(mean number)} = \frac{\sum_{i=1}^{N} (C_{ii} EN_i)}{\left(\sum_{i=1}^{N} EN_i\right)^2}$$

$$\text{Sojourn time} = \sum_{i=1}^{N} V_i (EW_i + \tau_i)$$

Example 3 1

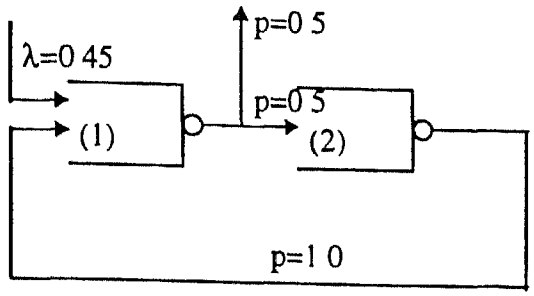


Figure 3 1 The open network for Example 3 1

Description of the network for Example 3 1

We have a two node single class open network There are external arrivals only to node 1 The interarrival time distribution for this is hyperexponential with mean 0 45 and SqCVar 2 25 Node 1 has hyperexponential service time distribution with mean 1 0 and SqCVar 2 25 Node 2 has Erlangian 4 service time distribution with mean 1 0 and SqCVar 0 25

Input parameters to Example 3 1

	Node 1	Node 2
m_i	1	1
λ_{0i}	0 45	0 0
C_{0i}^2	2 25	1 0
τ_i	1 0	1 0
C_i^2	2 25	0 25
f_i	1 0	1 0

Table 3 1 Input parameters of Example 3 1

	Node 1	Node 2
Node 1	0 0	0 5
Node 2	1 0	0 0

Table 3 2 Transition probability matrix of Example 3 1

Output parameters for Example 3 1 from QNAT

	Node 1	Node 2
λ_{0i}	0 45	0 0
C_{0i}^2	2 25	1 0
λ_i	0 90000	0 45
C_{ai}^2	1 749342	1 577437
EN_i	17 09734	0 786415
C_{si}^2	0 3078830	2 405392
EW_i	17 99704	0 747588
C_i^2	0 3056561	4 12567
dep_i	0 45	0
V_i	2 0000	1 0000
U_i	0 9000	0 4500

Table 3 3 Output parameters of Example 3 1

3 1 2 Fork Join Analysis (Open Networks)

The performance of distributed and parallel processing systems can be modelled by fork join analysis In this analysis the sibling queues can be with or without synchronizing queues If there are k sibling queues then each job will be split to k subjobs and the subjobs are served independently of one another The jobs will recombine when all the subjobs complete service in the sibling queues In the case of fork join queues without synchronizing queue the jobs will recombine when the last subjob finishes its service at its sibling queue In other words recombination of the subjobs will be followed by the departure of the subjob coming out of the sibling queue that has minimum service rate i e the subjob spends maximum time in that sibling queue Thus the effective service time distribution of the sibling process will be maximum of the k random service times

Assumptions

- All the sibling queues will have infinite buffer capacity
- The service discipline at all the sibling queues are FCFS

- There will be no external arrivals into the sibling queues (λ_{0i})

  $\lambda_{0i} = 0\ 0$ for $i = 1\ 2 \quad k$

  There will be neither creation nor combination of jobs at the sibling queues

  The multiplication factor (γ) is 1 0 at the resultant single queue $\gamma = 1\ 0$
- All the sibling queues are single server queues
- The service time distribution at all the sibling queues are exponential

Input parameters

k number of siblings in the fork-join queue

$1/\mu_i$ mean service time at sibling queue i for $i = 1\ 2 \quad k$

Output parameters

τ mean resultant service time distribution of the fork join model

C_s^2 squared coefficient of variance of resultant service time distribution

Algorithm 3 2 Fork join without synchronizing queues for open networks

$1/\mu_i$ for $i = 1\ 2 \quad k$

$F_{si}(x) = 1 - e^{-\mu_i x}$ (Exponential distribution with mean μ_i for queue i) for $i = 1\ 2 \quad k$

$f_{si}(x) = \mu_i e^{-\mu_i x}$ (Density corresponding to above distribution) for $i = 1\ 2 \quad k$

$F_w(x) = \prod_{1}^{K} F_i(x)$ (Service time distribution = maximum of k exponentials)

$f_w(x) = \sum_{i=1}^{K} \left(f_{si}(x) \prod_{\substack{j \\ j \neq i}}^{K} F_{sj}(x) \right)$ (Density corresponding to above distribution)

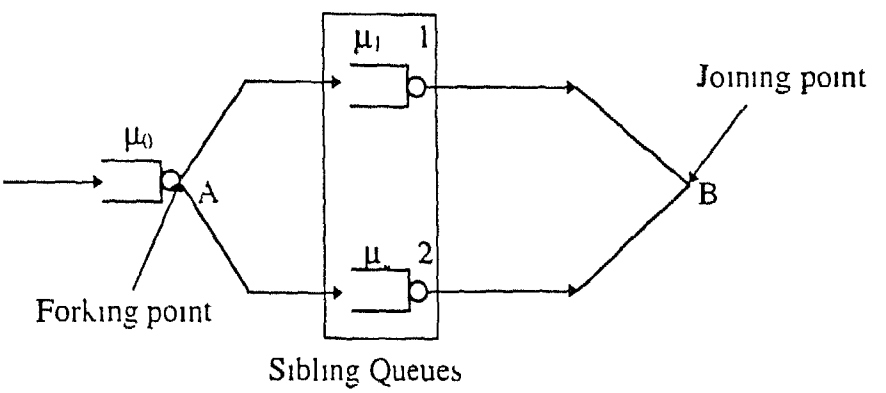
(Algorithm 3 2 continued)

$$\tau = \int_0^{\infty} f_w(x)\, x\, dx \quad \text{(Mean of the } F_w(x)\text{)}$$

$$\tau^2 = \int_0^{\infty} f_w(x)\, x^2\, dx \quad \text{(Second moment } F_w(x)\text{)}$$

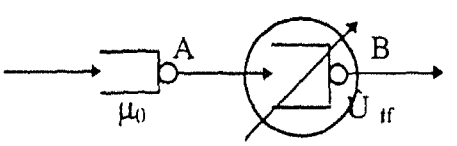
$$C_s^2 = \frac{\tau^2}{(\tau)^2} - 1 \quad \text{(Squared coefficient of variance of } F_w(x)\text{)}$$

Example 3 2



Sibling Queues

a Fork join without synchronizing queues in open network



b Equivalent reduced network

Figure 3 2 A fork join queue without synchronizing queues in an open network for Example 3 2

Description of the network for Example 3 2

There are two sibling queues without synchronizing queues in the sibling process The mean service rate at sibling queue 1 is μ_1 and in sibling queue 2 is μ as shown in Figure 3 2 At both the sibling queues the service time distribution is exponential The equivalent reduced network is then obtained as shown in Figure3 2b This open network can then be solved using Algorithm 3 1

Input parameters to Example 3 2

	Sibling queue 1	Sibling queue 2
$1/\mu_i$	20	10

Table 3 4 Input parameters of Example 3 2

Output parameters for Example 3 2 from QNAT

	Resultant queue
τ	23 33333
C_s^2	0 673469

Table 3 5 Output parameters of Example 3 2

For a fork join queue with synchronizing queues after completion of service at the sibling queues the jobs will stay in the synchronizing queues When all the subjobs finish service only then are the jobs recombined Analysis techniques for such queues are not available in the literature Therefore this model has not been implemented in QNAT

3 1 3 Closed Networks

A closed network is one in which jobs are circulated inside the network There will be no external arrivals to any of the nodes or external departures from any of the nodes in the network The algorithm used for the analysis of such closed networks with single class input is state dependent MVA[1 7] This is an iterative algorithm which requires only the mean service time (first moments) of the service time distributions

Assumptions

- All the queues in the network have infinite buffer capacity
- There are no external arrivals to any of the nodes in the network

[1] Edward D Lazowska John Zahorjan G Scott Graham and Kenneth C Sevcik *Quantitative System Performance* Prentice Hall Inc 1984
Michael K Molloy *Fundamental of Performance Modelling* Macmillan Publishing Company 1989

- The routing probabilities for the jobs in the network are static
- The service time distribution of the queues in the network are exponential

Input parameters

N number of nodes in the network

m_i number of servers at queue i for $i = 1\ 2 \quad ,N$

τ_i mean service time at queue i for $i = 1\ 2 \quad N$

$Qtype_i$ service discipline at queue i for $i = 1\ 2 \quad ,N$

FCFS LCFS PR PS and IS

[P] p_{ij} routing matrix $i\ j = 1\ 2 \quad ,N$

Output parameters

EN_i mean number at queue i for $i = 1\ 2 \quad N$

EW_i mean waiting time at queue i for $i = 1\ 2 \quad N$

λ_i mean throughput at queue i for $i = 1\ 2 \quad N$

V_i visit ratio at queue i for $i = 1\ 2 \quad N$

Algorithm 3 3 State dependent MVA solution technique for closed networks

M number of customers in the closed network
N number of nodes in the network
τ_{ij} mean service at queue i when j in the queue for $i = 1\ 2 \quad N$
for $j = 1\ 2 \quad M$

$= \tau_i$ if $Qtype\ i$ is IS

$$= \begin{cases} \dfrac{\tau_i}{j} & \text{if } j < m_i \\ \dfrac{\tau_i}{m_i} & \text{if } j \geq m_i \end{cases}$$

for $i = 1\ 2 \quad N$
for $j = 1\ 2 \quad M$
$Qtype\ i$ = FCFS LCFS PR or PS

(Algorithm 3 3 continued)

The visit ratios are obtained by solving the set of linear equations

$$V_i = \sum_{j=1}^{N} V_j \, p_{ji} \qquad \text{for } i = 1\ 2 \quad ,N$$

Initialisation

$N_i(0) = 0$ for $i = 1\ 2$ $,N$

$jmp_i(0\ 0) = 1$ for $i = 1\ 2$ N

$jmp_i(j\ 0) = 0$ for $j = 1\ 2$ M for $i = 1\ 2$ $,N$

Do steps 1 2 and 3 for $c = 1\ 2$ M

Step 1

$$w_i(c) = \sum_{j=1}^{c} j\, \tau_i(j)\, jmp_i(j\ 1\ c\ 1) \qquad \text{for } i = 1\ 2 \quad N$$

$$W(c) = \sum_{1}^{N} V_i\, w_i(c)$$

$$\lambda(c) = \frac{c}{W(c)}$$

Step 2

$N_i(c) = \lambda(c)\, V_i\, w_i(c)$ for $i = 1\ 2$ $,N$

Step 3

$jmp_i(j\ c) = \lambda(c)\, V_i\, \tau_i\, jmp_i(j\ 1\ c)$ for $i = 1\ 2$ $,N$ and for $j = 1\ 2$ c

$$jmp_i(0\ c) = 1 - \sum_{j=1}^{c} jmp_i(j\ c) \qquad \text{for } i = 1\ 2 \quad ,N$$

$EN_i = N_i(M)$ $EW_i = w_i(M)$ $\lambda_i = \lambda(M)\, V_i$ for $i = 1\ 2$ N

Example 3 3

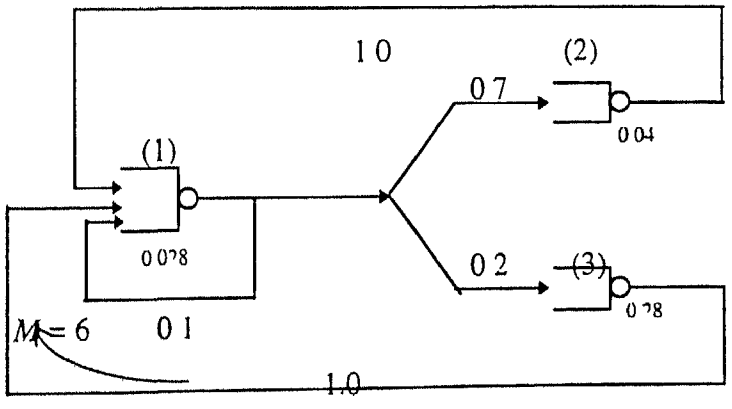


Figure 3 3 Closed network with a single class of customers for Example 3 3

Description of the network for Example 3 3

There are 3 nodes and 6 jobs in the network Node 1 uses a PS service discipline Node 2 and Node 3 use the FCFS service discipline This example is taken from Molloy[1] This is a model of a simple computer system with Node 1 as the processor and Nodes 2 and 3 as the I/O devices The service time distributions at all nodes are exponential The mean service time at Nodes 1 2 and 3 are 0 028 0 04 and 0 28 respectively

Input parameters to Example 3 3

	Node 1	Node 2	Node 3
m_i	1	1	1
τ_i	0 028	0 040	0 280
$Qtype_i$	PS	FCFS	FCFS

Table 3 6 Input parameters of Example 3 3

	Node 1	Node 2	Node 3
Node 1	0 1	0 7	0 2
Node 2	1 0	0 0	0 0
Node 3	1 0	0 0	0 0

Table 3 7 Routing matrix for Example 3 3

Output parameters for Example 3 3 from QNAT

	Node 1	Node 2	Node
EN_i	0 886640	0 886640	4 22672
EW_i	0 02310	0 03300	0 93800
λ_i	17 35107	12 14575	3 47021
V_i	1 00000	0 70000	0 20000

Table 3 8 Output parameters of Example 3 3

[1] Michael K Molloy *Fundamental of Performance Modelling* Macmillan Publishin Company 1989

3 1 4 Fork Join Analysis (Closed Networks)

In the case of a fork join queue without synchronizing queue in a closed queueing network the jobs will be combined after the last subjob has finished the service at its sibling queue In this the reduction step is a two fold one The first step is the same as the one stated in Section 3 1 2 for the case without synchronizing queue i e getting the resultant distribution of the service time In the second step we fit an exponential distribution to this resultant distribution such that the mean square error between the two distributions is minimum

Assumptions

- All the sibling queue have infinite buffer capacity
- The service discipline is FCFS at all the sibling queues
- The service time distribution at each sibling queue is exponential
- Each sibling queue is a single server queue
- There are no external arrival to any of the sibling queues in the model

Input parameters

k number of sibling queue in the fork join model

$1/\mu_i$ mean service time at the sibling queues $i = 1\ 2 \quad k$

Output parameters

τ mean resultant service time of the fork join model (exponential fit)

Algorithm 3 4 Fork join without synchronizing queues for closed networks

$1/\mu_i$ for $i = 1\ 2 \quad k$

$F_{si}(x) = 1 - e^{-\mu_i x}$ (Exponential distribution with mean μ_i for queue i) for $i = 1\ 2 \quad k$

$f_{si}(x) = \mu_i e^{-\mu_i x}$ (Density corresponding to above distribution) for $i = 1\ 2 \quad k$

(Algorithm 3 4 continued)

$$F_w(x) = \prod_{i=1}^{K} F_{si}(x)$$ (Service time distribution = maximum of k exponentials)

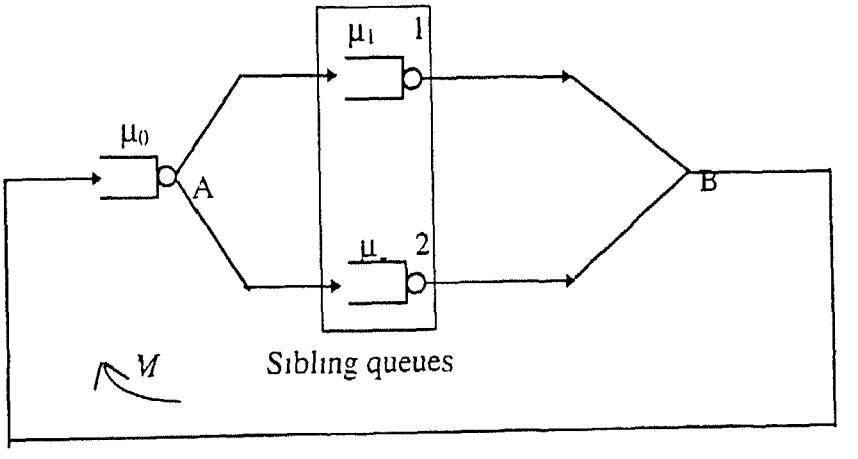
$$F_{estm}(x) = 1 - e^{-\hat{U}x}$$ (An exponential fit to $F_w(x)$)

$$\text{error}(x, \hat{U}) = \int_0^{\infty} [F_w(x) - F_{estm}(x)]^2 dx$$

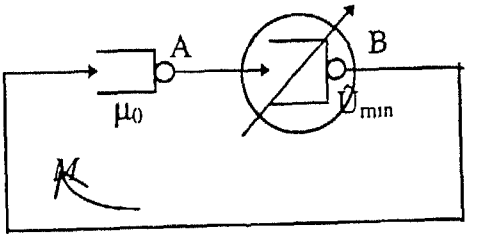
$$\hat{U}_{min} = \min_{\hat{U}} [\text{error}(x, U)]$$

$$\tau = 1/U_{min}$$

Example 3 4



a A network with a fork join queue without synchronizing queues



b The equivalent reduced network

Figure 3 4 Fork join queues without synchronizing queues in a closed network for Example 3 4

Description of the network for Example 3 4

There are two sibling queues in the sibling process There are no synchronizing queue for either of the sibling queues The mean service rate at sibling queue 1 is μ_1 and sibling queue 2 is μ_2 At both the sibling queues the service time distribution is exponential The resultant service time distribution is also exponential The equivalent network in Figure 3 4b is a closed network and can be solved by using Algorithm 3 3

Input parameters to Example 3 4

	Sibling queue 1	Sibling queue 2
$1/\mu_i$	20	10

Table 3 9 Input parameters of Example 3 4

Output parameters for Example 3 4 from QNAT

	Resultant queue
τ	23 64108

Table 3 10 Output parameters of Example 3 4

In the case of fork join with synchronizing queues after completion of service at the sibling queue the subjobs will wait in the synchronizing queues When all the subjobs finish their service they are recombined The solution procedure for such a queue is based on decomposition and aggregation This algorithm is from[1] There are two procedures to analyse this situation One is a numerical procedure which can be applied for small system ($\approx$ 2 sibling queues) In this procedure the following steps are carried out

- Generation of all states of the system and the corresponding rate matrix Q
- Solution of the linear system $\pi Q_s^T = 0 \Rightarrow$ steady state probability distribution

[1] Y C Liu and H G Perros *A Decomposition for the Analysis of a Closed Fork/Join Queueing System* IEEE Trans on Computers Vol 40 No 3 pp 365 370 Mar 1991
Y C Liu and H G Perros *Approximate Analysis of a Closed Fork/Join model* European Journal of Operational Research 53 pp 382 392 1991

The second approach is an approximate procedure which can be applied for large systems (≥ 2 sibling queues) For such networks we first obtain the FESC for the fork join queue by shorting the input and the output of the fork join node This FESC is then used in the larger network which is solved using techniques described before The FESC for the fork join queue is then obtained as follows

- Two siblings are selected and solved exactly to obtain their FESC
- This FESC is used to successively aggregate the other k 2 siblings into the FESC

Assumptions

- All the sibling queue have infinite buffer capacity
- The service discipline is FCFS at each of the sibling queues
- The service time distribution at each sibling queue is exponential
- Each sibling queue is a single server queue
- There are no external arrival to any of the sibling queues in the model

Input parameters

k number of sibling queues in the fork join model

M number of customers(jobs) in the network

$\tau_i(j) = 1/\mu_i(j)$ mean service time at sibling queue i when j customers in it

Output parameters

$\tau_{eff}(j)$ resultant mean service time of the fork join mode with j jobs in it

Algorithm 3 5 Fork join with synchronizing queues for closed networks

Do steps 1 & 2 for $c = 1\ 2 \quad k\ 1$
Step1 Do (A) (B) and (C) for $j = 1\ 2 \quad ,M$

(A) $P_{nj} = P_{jj} \prod_{k-+1}^{j} \rho_{c+1}(k) \qquad 0 \leq n \leq j\ 1$

(Algorithm 3 5 continued)

(B) $P_{jn} = P_{jj} \prod_{k=+1}^{j} \rho_c(k) \qquad 0 \le n \le j\ 1$

(C) $P_{jj} = \dfrac{1\ 0}{1 + \sum_{-0}^{j\ 1} \prod_{q=+1}^{j} \rho_{c+1}(q) + \sum_{i\ 0}^{j\ 1} \prod_{l\ i\ 1}^{j} \rho_c(l)}$

where

$$\rho\ (i) = \frac{\mu_{c+1}(i)}{\mu_c(j)} \qquad \rho_{c+1}(j) = \frac{\mu_c(i)}{\mu_{c+1}(j)} \qquad \text{for } i = 1\ 2 \quad j$$

Step2

$$\mu_{c+1}(j) = \mu_{c+1}(j) \sum_{-0}^{j-1} P_{ij} + \mu_c(j) \sum_{=0}^{j-1} P_j$$

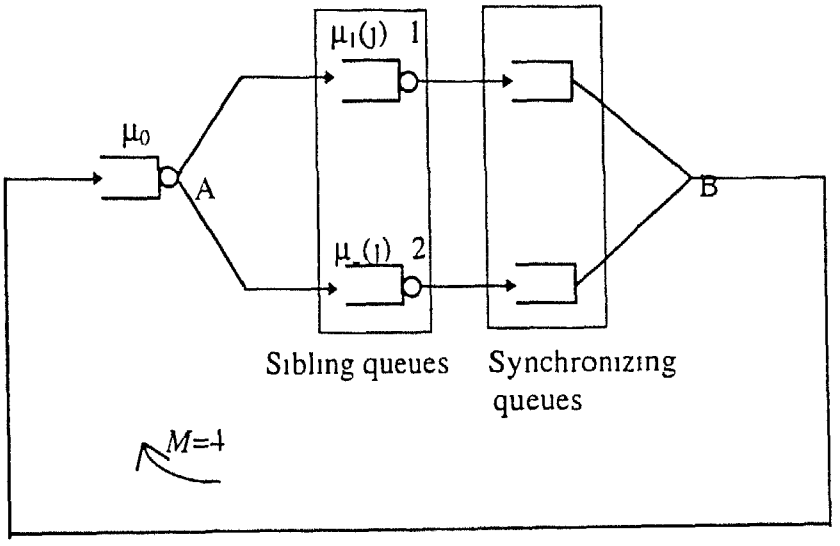
$$\tau_{eff}(j) = 1/[\mu_{A\ 1}(j)] = 1/[U_{etf}(j)] \quad j = 1\ 2 \qquad M$$

The GUI in QNAT allows us to input the service time at the sibling queues as a table or as an expression For the case of table the mean service time at each sibling queue is entered separately for each value of the number of jobs at that sibling queue Example 3 5 will illustrate a situation where the mean service time input is given as a table For the case of an expression the mean service time is entered as a function of n the number of customers in the sibling queue The expression can be a constant or a function of the variable n in the standard mathematical form Example 3 6 will illustrate a situation where the mean service time is input as an expression

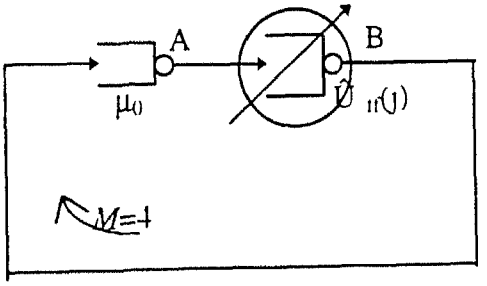
Description of the network for Example 3 5

There are two sibling queues with synchronizing queues in the closed network with four customers in the network jobs at point A in Figure 3 5a will be split into two and passed on to the sibling queues After service at the sibling queues the subjobs will wait in the respective synchronizing queues When both the subjobs finish at their respective sibling queues they recombine at point B in Figure 3 5a The service rate at both the sibling

queues are state dependent and have an exponential distribution This service can be given in two forms either as a table or as an expression The equivalent network in Figure 3 5b is like a two node network with service centres having state dependent service rate This may be analyzed using Algorithm 3 3 to get the performance of the equivalent network



a Fork join with synchronizing queues in a closed network



b Equivalent reduced network

Figure 3 5 A fork join queue with synchronizing queues in a closed network for Examples 3 5 and 3 6

Example 3 5 Service time as a Table

Input parameters to Example 3 5

Number	1	2	3	4
Sibling queue1	10 5	10 25	10 125	10 0625
Sibling queue2	5 66667	5 44444	5 296296	5 19753

Table 3 11 Input parameters of Example 3 5

Output parameters for Example 3 5 from QNAT

Number	1	2	3	4
	12 486	10 683	10 238	10 092

Table 3 12 Output parameters of Example 3 5

Example 3 6 Service time as an Expression

Input parameters to Example 3 6

Sibling queue1	10 + 2 ^ (n)
Sibling queue2	5 + 1 5 ^ (n)

Table 3 13 Input parameters of Example 3 6

Output parameters for Example 3 6 from QNAT

Number	1	2	3	4
	12 486	10 683	10 238	10 093

Table 3 14 Output parameters of Example 3 6

3 2 Queueing Networks with Multiple Classes of Jobs

When there are more than one class of customers in the network the system can be open with respect to some classes of customers and closed with respect to the other classes If the number of closed class customers is zero than the system is an open network If the number of open classes is zero then the system is a closed network Otherwise the network is called a mixed network

In developing a queueing network model of a system we may need to allow a customer to change classes after service at a queue and be routed according to the routing probabilities of that class It has been shown that such a general model can be trivially mapped into a BCMP network that does not allow change of class Therefore in QNAT we do not allow customers to change classes

3 2 1 Open Networks

The network is considered to be an open network when all the classes in the network are open In the case of multiple classses of jobs there must be external arrivals/departures to/from the network for each class This part has been earlier implemented by Umesh and the details of the algorithm and its implementation can be obtained from [1]

3 2 2 Closed Networks

In a closed network all the jobs of all the different classes are to be circulated inside the system There will be no external arrivals to the system and hence no departures out of the system The technique used to evaluate the closed queueing network is the Mean Value Analysis (MVA) technique The parameters like mean queue length mean waiting time mean throughput etc in closed multiple class queueing network which have product form solution can be computed recursively without computing the product terms and the normalizing constants However from this technique we can only get the first moments In spite of this limitation we have decided to use this algorithm because of the simplicity of its implementation

This algorithm is based on the Mean Value Theorem [3 4] This theorem states that a customer of class k arriving to a queue in the network would see the same average

---

[1] Ward Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 198
M N Umesh *A Window based Package for Queueing Network Analysis* M Tech thesis IIT Kanpur Feb 1997

[3] M Reiser and S S Lavenberg *Mean Value Analysis of Closed Multichain Queueing Networks* J of ACM Vol 27 No 2 pp 313 322 April 1980

[4] Michael K Molloy *Fundamental of Performance Modelling* Macmillan Publishing Company 1989

number in the queue as an external observer would see if the network had one less customer of class k The concept of visit ratio is used in the algorithm Visit ratios are defined as the ratios of throughputs Visit ratio at node k V_k is the ratio of throughput of node k to the throughput of the reference node in the network For the analysis of closed networks node 1 is arbitrarily selected as the reference node in QNAT to calculate the visit ratios It is easy to see that $V_k = \lambda_k / \lambda_1$ where λ_i is the absolute arrival rate to node i In mean value analysis we first initialise and compute the waiting time at each node for each class in the network Next we compute the throughput of each class followed by the mean queue length at each node for each class in the network Finally we update the queue length distribution These parameters are computed iteratively until we reach the total population of each class

Assumptions

- All the queues in the network have infinite buffer capacity
- There are no external arrivals to any of the queues for any of the classes
- Classes are independent and the routing probabilities for each class in the network are static
- Exponential service times for all classes at all nodes in the network
- Only the following four service discipline are considered in the network FCFS LCFS PR PS and IS and are indicated in the GUI as follows

  FCFS 1 LCFS PR 2 PS 3 IS 4
- There is at least one class of jobs with a non zero population

Input parameters

N number of nodes in the network

R total number of classes

m_i number of servers at node i for $i = 1\ 2 \ldots N$

τ_{ik} mean service time at node i of class k for $i = 1\ 2 \ldots N$ & $k = 1\ 2 \ldots R$

$Qtype_i$ service discipline at node i for $i = 1\ 2 \ldots N$

$\underline{C}$ population vector = $\{C_1\ C_2 \ldots C_R\}$ where C_k is the population of class k

$[Q_k]_{N \times N}$ $q_{k(i,j)}$ routing matrix for $i\ j = 1\ 2 \ldots N$ & $k = 1\ 2 \ldots R$

Output parameters

EN_{ik} mean number of class k customers at node i for $i = 1\ 2 \quad N$ & $k = 1\ 2 \quad R$

EW_{ik} mean waiting time of class k customers at node i for $i = 1\ 2 \quad N$

$k = 1\ 2 \quad R$

λ_{ik} mean throughput of class k customers at node i for $i = 1\ 2 \quad N$

$k = 1\ 2 \quad R$

V_{ik} visit ratio of class k customers at node i for $i = 1\ 2 \quad N$ & $k = 1\ 2 \quad R$

$$EN_i = \sum_{k=1}^{R} EN_{ik} \quad \text{for } i = 1\ 2 \quad N$$

Algorithm 3 6 Multiple class MVA solution technique for closed networks [1]

N number of nodes in the network
R number of classes
$\underline{C}$ $\{C_1\ C \quad C_R\}$ population vector where C_k is total population of class k
$\underline{0} = \{0\ 0 \quad 0\}$ zero vector zero in each class for all R classes
$\underline{e}_k = \{0\ 0 \quad 1\ 0 \quad 0\}$ unit vector in the k^{th} position

1) Initialisation

$LN_i(\underline{0}) = 1\ 0$ for $i = 1\ 2 \quad N$
$P_i(0\ \underline{0}) = 1\ 0$ for $i = 1\ 2 \quad N$
$P_i(j\ \underline{0}) = 0\ 0$ for $i = 1\ 2 \quad N$

2) Do steps 3 4 & 5 for $c_1 = 0\ 1\ 2 \quad C_1$ $c_2 = 0\ 1\ 2 \quad C$ $c_k = 0\ 1\ 2 \quad C_k$

3) Do for $k = 1\ 2 \quad R$ & $i = 1\ 2 \quad N$

$$w_{ik}(\underline{C}) = \begin{cases} \tau_{ik}\, V_{ik}\, LN(\underline{C}) \\ \tau_{ik}\, V_{ik} \\ \tau_{ik}\, V_{ik}\, (LN_i(\underline{C}) + \sum_{j\ 1}^{1} (m_i\ j) P_i(j \mid \underline{C} \quad \underline{e}_k) \end{cases}$$

$$W_k(\underline{C}) = \sum_{1}^{N} w_{ik}(\underline{C})$$

[1] M Reiser and S S Lavenberg *Mean Value Analysis of Closed Multichain Queueing Networks* J of ACM Vol 27 No 2 pp 313 322 April 1980
[2] Michael K Molloy *Fundamental of Performance Modelling* Macmillan Publishing Company 1989

(Algorithm 3 6 continued)

$$\lambda_k(\underline{C}) = \frac{c_k}{W_k(\underline{C})}$$

4) $LN_i(\underline{C}) = 1 + \sum_{k\ 1}^{R} \lambda_k(\underline{C})\, V_{ik}\, w_{ik}(\underline{C})$ for $i = 1\ 2 \quad ,N$

5) Do for $i = 1\ 2 \quad ,N$ & $j = 1\ 2 \quad m_i\ \ 1$

$$P_i(j\ \underline{C}) = (1/j) * \sum_{k-1}^{R} \lambda_k(\underline{C})\, V_{ik}\, \tau_{ik}\, P_i(j\ 1\ \underline{C} - \underline{e}_k)$$

$$P_i(0\ \underline{C}) = 1 \quad \frac{\sum_{k-1}^{R} \lambda_k(\underline{C})\, V_{ik}\, \tau_{ik} \ + \ \sum_{j\ 1}^{-1} (m_i\ \ j)\, P_i(j\ \underline{C})}{m_i}$$

$\underline{C}\ \ \{ C_1\ C \qquad C_R \}$

$EN_i = LN_i(\underline{C})\ \ 1$ for $i = 1\ 2 \quad .N$

$EN_{ik} = \lambda_k(\underline{C})\, w_{ik}(\underline{C})$ for $i = 1\ 2 \quad N$ & $k = 1\ 2 \quad R$

$$EW_{ik} = \frac{w_{ik}(\underline{C})\ \ V_{ik}\, \tau_{ik}}{V_{ik}} \qquad \text{for } i = 1\ 2 \quad .N \ \&\ k = 1\ 2 \quad R$$

$\lambda_{ik}(\underline{C}) = \lambda_k(\underline{C})\, V_{ik}$ for $i = 1\ 2 \quad N$ & $k = 1\ 2 \quad R$

Example 3 7

Description of the network for Example 7

There are 2 classes and 3 nodes in the closed network There are a total of 6 customers with 3 in each class The service discipline for both the classes at node 1 is PS and at node 2 and node 3 are FCFS The mean service time for both the classes at node 1 is 0 028 node 2 is 0 040 and node 3 is 0 280 This is similar to the one given in Example 3 3

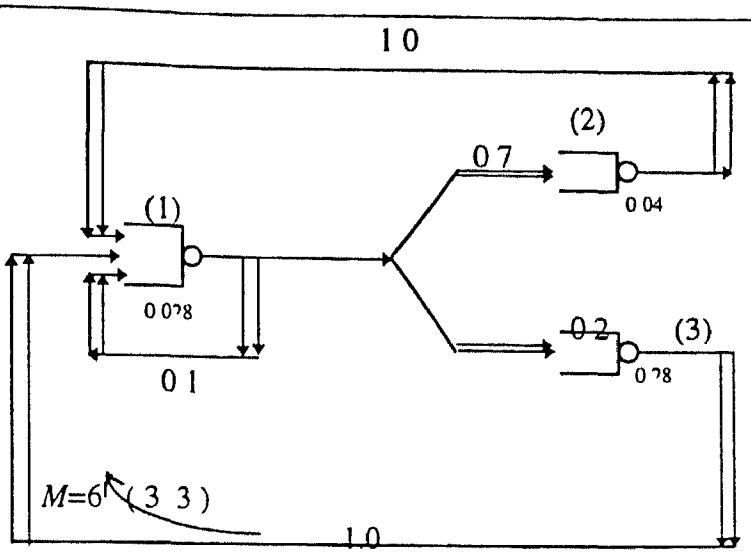


Figure 3 6 Multiple class Closed Network(2 classes) for Example 3 7

Input parameters to Example 3 7

	Node 1	Node 2	Node 3
m_i	1	1	1
$Qtype_i$	3	1	1

Table 3 15 Input parameters of Example 3 7

τ_{ik}	Node 1	Node 2	Node 3
Class 1	0 028	0 040	0 280
Class 2	0 028	0 040	0 280

Table 3 16 Mean service time of Example 3 7

	Node 1	Node 2	Node 3
Node1	0 1	0 7	0 2
Node 2	1 0	0 0	0 0
Node 3	1 0	0 0	0 0

Table 3 17 Routing probabilities for class1 customers of Example 3 7

	Node 1	Node 2	Node 3
Node 1	0 1	0 7	0 2
Node 2	1 0	0 0	0 0
Node 3	1 0	0 0	0 0

Table 3 18 Routing probabilities for class2 customers of Example 3 7

Output parameters for Example 3 7 from QNAT

	Node 1		Node 2		Node 3	
EN_i	0 886640		0 886640		4 22672	
classes	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
EN_{ik}	0 44332	0 44332	0 44332	0 44332	2 113360	2 11336
EW_{ik}	0 02310	0 2310	0 3300	0 3300	0 93800	0 93800
$\lambda_{ik}(\underline{C})$	8 67553	8 67553	6 07287	6 07287	1 735107	1 735107
V_{ik}	1 0000	1 0000	0 7000	0 7000	0 2000	0 2000

Table 3 19 Output parameters of Example 3 7

3 2 3 Mixed Networks

Mixed networks are those in which some classes are open and some are closed with respect to the system The open classes in the network are characterised by the total arrival rate of that class and the closed classes are characterised by the population vector (population of each closed class) The algorithm is taken from the *Lazowska et al* [1] In this algorithm the utilizations of the open classes at each node are computed first followed by the nett utilization of each node due to the open classes The performance parameters of the closed classes are computed by eliminating the open classes and converting the mixed model into a closed model with inflated service demands for the closed classes The inflation factor used is 1 $U_{i(O)}$ (1 utilisation due to open classes at node i) and $U_{i(O)}$ is given by

$$U_{i(O)} = \sum_{c}^{O} U_{ic} \qquad \text{where } U_{ic} \text{ is the utilisation at node } i \text{ of class } c$$

[1] Edward D Lazowska John Zahorjan G Scott Graham and Kenneth C Sevcik *Quantitative System Performance* Prentice Hall Inc 1984

This is the percentage of time the server is not used by open classes and is the time available for the closed classes to get service from the server in the queue The performance of the closed classes is analysed using the standard solution techniques discussed earlier with the inflated service times for the closed class customers The performance of the open classes are computed by considering the mean queue length of the closed classes at each node in the network The GUI permits the use of zero open class with non zero closed classes or vice versa

Assumptions

- All the queues in the network have infinite buffer capacity
- The service discipline for all classes are FCFS at all the nodes in the network
- There can be creation or combination of jobs for the open classes
- All the classes are independent
- There must be an external arrival for each of the open classes These processes must be Poisson
- The service time distribution for all the classes in the network are exponential
- The routing chain for the jobs in the network are static

Input parameters

R total number of classes in the mixed model

O total number of open classes in the mixed network

C total number of closed classes in the mixed network

$R = O + C$

N number of nodes in the mixed model

m_i number of servers at node i in the mixed network for $i = 1\ 2 \quad N$

$\underline{C}$ population vector of the closed classes

$= \{ C_1\ C \qquad C_C \}$

$[Q_k]_{N \times N}\ q_{k(i,j)}$ routing probabilities for $i, j = 1\ 2 \quad N$ & $k = 1\ 2 \quad R$

Open classes

$\lambda_{0i,c}$ mean arrival rate of class c at node i for $c = 1\ 2 \quad O$ & $i = 1\ 2 \quad N$

τ_{ic} mean service time distribution of class c at node i $c \in \{ O \}$

for c=1 2 O & i=1 2 N

γ_{ic} multiplication factor of class c at node i for i =1 2 N & c=1 2 O

Closed classes

τ_{ic} mean service time distribution of class c at node i $c \in \{ C \}$

for c=1 2 C & i=1 2 N

Output parameters

EN_{iopen} mean open class jobs at node i

$EN_{i\ closed}$ mean closed class jobs at node i

Open classes

λ_i nett arrival of class c at the node i for i=1 2 N & c=1 2 O

EN_{iopen} mean queue length of class c at the node i for i=1 2 N

c=1 2 O

EW_{iopen} mean delay of class c at the node i for i=1 2 N

c=1 2 O

$dep_{iopen\,c}$ departure rate of class c at the node i for i=1 2 N

c=1 2 O

V_{iopen} visit counts of class c at the node i for i=1 2 N

c=1 2 O

$U_{i\,c}$ utilisation of class c at the node i for i=1 2 N & c=1 2 O

Closed classes

$EN_{i\ closed\,c}$ mean queue length of class c at node i for i=1 2 N

c=1 2 C

$EW_{i\ closed}$ mean delay of class c at node i for i=1 2 N & c=1 2 C

$\lambda_{ic\,closed\,c}$ throughput of class c at node i for i=1 2 N & c=1 2 C

$V_{i\ closed\,c}$ visit ratios of class c at node i for i=1 2 N & c=1 2 C

Algorithm 3 7 Mixed network solution technique [1]

R total number of classes
O total number of open classes
C total number of closed classes
N number of nodes in the network
$\{ O \}$ set of open classes
$\{ C \}$ set of closed classes

1) Solving the equation below we get the nett arrivals to the nodes of open classes

$$\lambda_{ik} = \lambda_{0i\,k} + \sum_{j=1}^{N} \lambda_{ik}\, q_{k\,(j\,i)} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad O$$

$$U_{i\,k} = \frac{\lambda_{ik}\,\tau_{ik}}{m_i} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad O$$

$$U_{i\,(O)} = \sum_{k=1}^{O} U_{i\,k} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad O$$

$$\text{dep}_{i\text{open}\,k} = \lambda_{ik}\,\gamma_{ik}\,(\,1 \sum_{j=1}^{N} q_{k\,(i\,j)}\,) \quad \text{for } i=1\,2 \quad N \;\&\; k=1\,2 \quad O$$

$$\text{throughput}(k) = \sum_{i=1}^{N} \lambda_{ik} \quad \text{for } k=1\,2 \quad O$$

$$V_{i\text{open}\,k} = \frac{\lambda_{ik}}{\text{throughput}(k)} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad O$$

2) Inflate the service time of the closed classes by a factor of $1 \quad Y_{i\,(O)}$

$$\tau_{ik} = \frac{\tau_{ik}}{1 \quad U_{i\,(O)}} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad C \;\&\; k \in \{C\}$$

Now solve the closed network with the algorithm 3 6 (multiple class closed network MVA)

[1] Edward D Lazowska John Zahorjan G Scott Graham and Kenneth C Sevcik *Quantitative System Performance* Prentice Hall Inc 1984

(Algorithm 3 7 continued)

3) The waiting time and the queue length for the open classes are

$$EW_{i\text{open }k} = \frac{\tau_{ik}\, V_{i\text{open }k}\,(1 + \text{Qlength}_{i\{C\}})}{1 \quad U_{i\{O\}}} \quad \text{for } i=1\,2 \quad ,N \;\&\; k=1\,2 \quad O \;\&\; k \in \{O\}$$

where

$$\text{Qlength}_{i\{C\}} = EN_{i\ \text{losed}} \quad \text{for } i=1\,2 \quad ,N$$

$$= \sum_{k=1}^{C} EN_{i\text{closed }k} \quad \text{for } i=1\,2 \quad ,N \;\&\; k \in \{C\}$$

$$EN_{i\text{open }k} = \text{throughput(k)}\; EW_{i\text{open }k} \quad \text{for } i=1\,2 \quad N \;\&\; k=1\,2 \quad O \;\&\; k \in \{O\}$$

$$EN_{i\text{open}} = \sum_{k=1}^{O} EN_{i\text{open }k} \quad \text{for } i=1\,2 \quad ,N \;\&\; k \in \{O\}$$

Example 3 8

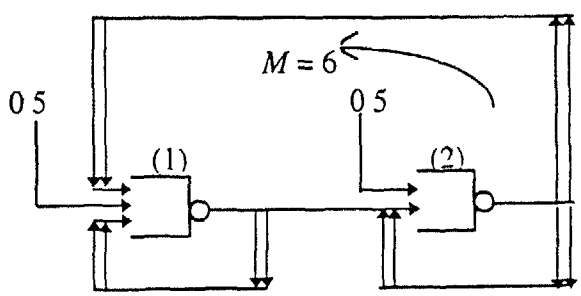


Figure 3 7 Multiple classes in Mixed Network (2 classes) (one open class and one closed class) for Example 3 8

Description of the network for Example 3 8

There are two nodes and two classes (one open class and one closed class) in the network Each open class is characterised by the total arrival rate (sum of arrival rate at node 1 and node 2) and each closed class is characterised by the total number of customers in the network (population of the closed class is 6) The service time distributions at all nodes for all classes are exponential The routing probabilities and the service times at each node for each class are given below

Input parameters to Example 3 8

Open class

	Nodes1	Node 2
$\lambda_{0i\,k}$	0 500	0 500
τ_{ik}	0 500	0 500
γ_{ik}	1 0	1 0

Table 3 20 Input parameters of open class in Example 3 8

Closed class

	Nodes1	Node 2
τ_{ik}	0 040	0 280

Table 3 21 Input parameters of closed class in Example 3 8

Routing probabilities

$q_{k\,ij}$	Nodes1		Node 2	
Classes	Class 1	Class 2	Class 1	Class 2
Node 1	0 1	0 2	0 5	0 8
Node 2	0 5	0 8	0 1	0 2

Table 3 22 Routing probabilities for Example 3 8

Output parameters for Example 3 8 from QNAT

Open class

	Node 1	Node 2
EN_{open}	1 944430	11 38890
$\lambda_{0i\,k}$	0 50	0 50
λ_{ik}	1 125	1 125
$EN_{iopen\,k}$	1 944430	11 38890
$EW_{iopen\,k}$	1 944430	11 38890
$dep_{iopen\,k}$	0 50	0 50
$V_{open\,k}$	1 125	1 125
U_{k}	0 6250	0 6250

Table 3 23 Output parameters of open class in Example 3 8

Closed class

	Node 1	Node 2
$EN_{iclosed}$	0 1666582	5 833340
$EN_{iclosed\ k}$	0 1666582	5 833340
$EW_{iclosed\ k}$	0 1777234	3 60893
$\lambda_{i\ losed\ k}$	1 339276	1 339276
$V_{iclosed\ k}$	1 0000	1 0000

Table 3 24 Output parameters of closed class in Example 3 8

Chapter 4

Summary and Future work

4 1 Summary

In this thesis we have developed a Microsoft Windows based software package for queueing network analysis called **QNAT** In this package some of the widely used solution techniques are used to solve a specified queueing network This package can be used to analyse a network of queues with queues having finite buffer capacity or infinite buffer capacity For infinite buffer capacity queues the jobs can be of a single class or of multiple classes To solve a single class open network we have used GI/G/m algorithm and to solve a closed network we have used the state dependent MVA algorithm QNAT also allows us to include fork join queues that may or may not have synchronising queue after the service queue in the siblings We have developed reduction techniques to handle fork join queues without synchronising queues for open and closed networks QNAT does not support fork join queues with synchronising queues in an open network

There are a few software packages that have already been developed to analyse queueing networks We overcome the major drawback in these packages by providing a use friendly GUI based front end to define the network and to view the results of the solution Also the use of Mathematica™ to carry out the computational tasks considerably simplifies the developement process and enables us to use mathematically involved algorithms Another salient feature of QNAT is that we have included many types of queueing networks in one package All the other packages that have been developed before restrict themselves to a small class of queueing networks

4 2 Future Work

There are still many models that are not yet supported in QNAT Some of these are Norton reduction change of job classes in the case of multiple classes permit multiple classes in fork join queues permit multiple servers in the fork join model nodes with finite capacity in the fork join queues and nodes with finite capacity in the networks with multiple classes jobs Apart form the mathematical analysis of a queueing network there is a need to provide simulation capability A simulation option when provided in QNAT can be used to define and execute a simulation model of the system under studys This simulation capability will also enable us to check the accuracy of the queueing network model We would also like to include discrete time queues and queueing networks into QNAT This is expected to be particularly important in modeling communication networks based on ATM technology Allowing a combination of continuous time queues and discrete time queues in a network would be an interesting challenge

Bibliography

1 Edward D Lazowska John Zahorjan G Scott Graham and Kenneth C Sevcik *Quantitative System Performance* Prentice Hall Inc 1984

2 F Baskett K M Chandy R R Muntz and F G Palacios *Open Closed and Mixed networks of queues with different class of customers* J of ACM Vol 22 No 2 pp 248 260 April 1975

3 K M Chandy U Herzog and L Woo *Approximate Analysis of General Queueing Networks* IBM Jour of Res and Develop 19 36 Jan 1975

4 Kim and A K Agrawala *Analysis of the Fork Join Queue* IEEE Trans on Computers Vol 38 No 2 pp 250 255 Feb 1989

5 Michael K Molloy *Fundamental of Performance Modelling* Macmillan Publishing Company 1989

6 M Chandy U Herzog and L Woo *Parametric Analysis of Queueing Networks* IBM Jour of Res and Develop 19 36 Jan 1975

7 N Umesh *A Window based Package for Queueing Network Analysis* M Tech Thesis IIT Kanpur Feb 1997

8 Reiser and S S Lavenberg *Mean Value Analysis of Closed Multichain Queueing Networks* J of ACM Vol 27 No 2 pp 313 322 April 1980

9 Ward Whitt *The Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983

10 W Whitt *Performance of the Queueing Network Analyser* B S T J Vol 62 No 9 Nov 1983

11 Whitt *Approximating a Point Process by a Renewal Process I Two Basic Methods* Operations Research Vol 30 No 1 pp 125 147 Jan Feb 1982

12 Y C Liu and H G Perros *A Decomposition for the Analysis of a Closed Fork/Join Queueing System* IEEE Trans on Computers Vol 40 No 3 pp 365 370 Mar 1991

13 Y C Liu and H G Perros *Approximate Analysis of a Closed Fork/Join model* European Journal of Operational Research 53 pp 382 392 1991

EE-1998-M-B11A-IMP